# Université de Montréal

# Better representation learning for TPMS

par


## Amir Raza

Département de mathématiques et de statistique
Faculté des arts et des sciences


Mémoire présenté en vue de l'obtention du grade de
Maître ès sciences (M.Sc.)
en Informatique

Orientation Intelligence Artificielle

October 8, 2021

# Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

**Better representation learning for TPMS**

présenté par

## Amir Raza

a été évalué par un jury composé des personnes suivantes :

*Jian-Yun Nie*
(président-rapporteur)

*Laurent Charlin*
(directeur de recherche)

*Golnoosh Farnadi*
(codirecteur)

*Jian Tang*
(membre du jury)

# Résumé

Avec l'augmentation de la popularité de l'IA et de l'apprentissage automatique, le nombre de participants a explosé dans les conférences AI/ML. Le grand nombre d'articles soumis et la nature évolutive des sujets constituent des défis supplémentaires pour les systèmes d'évaluation par les pairs qui sont cruciaux pour nos communautés scientifiques. Certaines conférences ont évolué vers l'automatisation de l'attribution des examinateurs pour les soumissions, le TPMS [1] étant l'un de ces systèmes existants. Actuellement, TPMS prépare des profils de chercheurs et de soumissions basés sur le contenu, afin de modéliser l'adéquation des paires examinateur-soumission.

Dans ce travail, nous explorons différentes approches pour le réglage fin auto-supervisé des transformateurs BERT pour les données des documents de conférence. Nous démontrons quelques nouvelles approches des vues d'augmentation pour l'auto-supervision dans le traitement du langage naturel, qui jusqu'à présent était davantage axée sur les problèmes de vision par ordinateur. Nous utilisons ensuite ces représentations d'articles individuels pour construire un modèle d'expertise qui apprend à combiner la représentation des différents travaux publiés d'un examinateur et à prédire leur pertinence pour l'examen d'un article soumis. Au final, nous montrons que de meilleures représentations individuelles des papiers et une meilleure modélisation de l'expertise conduisent à de meilleures performances dans la tâche de prédiction de l'adéquation de l'examinateur.

**Mots-clés** l'apprentissage de la machine, le traitement du langage naturel, des représentations textuelles, les transformateurs BERT, réglage fin, auto-surveillance, Apprentissage contrasté, évaluation par les pairs Automatiser, modélisation Expertise, prévision d'intérêt.

# Abstract

With the increase in popularity of AI and Machine learning, participation numbers have exploded in AI/ML conferences. The large number of submission papers and the evolving nature of topics constitute additional challenges for peer-review systems that are crucial for our scientific communities. Some conferences have moved towards automating the reviewer assignment for submissions, TPMS [1] being one such existing system. Currently, TPMS prepares content-based profiles of researchers and submission papers, to model the suitability of reviewer-submission pairs.

In this work, we explore different approaches to self-supervised fine-tuning of BERT transformers for conference papers data. We demonstrate some new approaches to augmentation views for self-supervision in natural language processing, which till now has been more focused on problems in computer vision. We then use these individual paper representations for building an expertise model which learns to combine the representation of different published works of a reviewer and predict their relevance for reviewing a submission paper. In the end, we show that better individual paper representations and expertise modeling lead to better performance on the reviewer suitability prediction task.

**Keywords** Machine learning, Natural language processing, Text representations, BERT transformer, Fine-tuning, Self-supervision, Contrastive learning, Automating peer-review, Expertise Modelling, Interest prediction.

# Contents

# List of Tables

# List of Figures

# Liste des sigles et des abréviations

| | |
|---|---|
| LDA | Latent Dirichlet Allocation |
| BOW | Bag of Words |
| TF-IDF | Term Frequency - Inverse Document Frequency |
| MSE | Mean Squared Error |
| BERT | Bidirectional Encoder Representations from Transformers |
| TPMS | Toronto Paper Matching System |
| MIP | Mixed Integer Program |
| BYOL | Bootstrap your own latent |
| SimCLR | Simple Framework for Contrastive Learning of Visual Representations |
| NeurIPS | Neural Information Processing Systems |

PLDI Programming Language Design and Implementation

# Remerciements

I would like to thank my supervisors Laurent Charlin and Golnoosh Farnadi for all their guidance. During the course of my research, I have learned how to ask the right questions, instead of being dazzled by the shine of new models. I feel that this was the most important part of my learning process, and has prepared me well to take on topics, which I may not have encountered yet. There were discussions week after week, and in spite of their busy schedules, and the corona pandemic which has tossed away all sense of normality, I felt very supported by both of my supervisors throughout this research process. I would also like to support my colleague, Meghana Moorthy Bhat, who was always eager to share many great ideas with me and was a valuable source of support and inspiration from time to time. And finally, I would thank my parents and my brother who were my pillars of support all through my journey of learning and explorations.

# Introduction

Academic conferences are venues to publish and discuss new research. Conferences can be focused on a specific field such as EMNLP or SIGGRAPH, or be multidisciplinary in nature such as FAccT.[1][2][3] Each conference aims to publish high-quality papers that adhere to the highest research standards, and maintain the research integrity. This relies on correctly evaluating paper submissions as that would attract more researchers for the future and improve the venue's reputation. Ultimately well managed conferences help science to progress.

Peer-review is the system adopted by academic conferences to judge the suitability of newly created research papers, evaluated by colleagues who act as expert referees. These referees, called reviewers play an important role in choosing promising work and giving constructive feedback for improvement.

# Problem setup

The number of submissions to machine learning and artificial intelligence (AI) conferences has increased each year due to increasing popularity of the field. Ideally, every submission ought to be reviewed by an expert on the topic of the submission, assigned by an impartial program committee. But tight timelines, increasing volume of research submissions and new evolving subfields, make manual assignment of reviewers impossible. All these can lead to conference organizers feeling overwhelmed. Reviewers and program chairs can find it challenging to provide appropriate feedbacks. Partial solutions include increasing the pool of suitable reviewers or improving the quality of the expert assignment, where each reviewer is assigned to review the most suitable submission paper.

Our research accepts the fact that automating conferences is the only way forward. Hence to improve the current process, we shall first discuss what is the existing process for an automated paper reviewer matching system. Once we have identified the individual steps, we will see which tools from ML can be used to improve them.

[1]EMNLP website https://2021.emnlp.org/
[2]SIGGRAPH website https://www.siggraph.org/
[3]FAccT website https://facctconference.org/

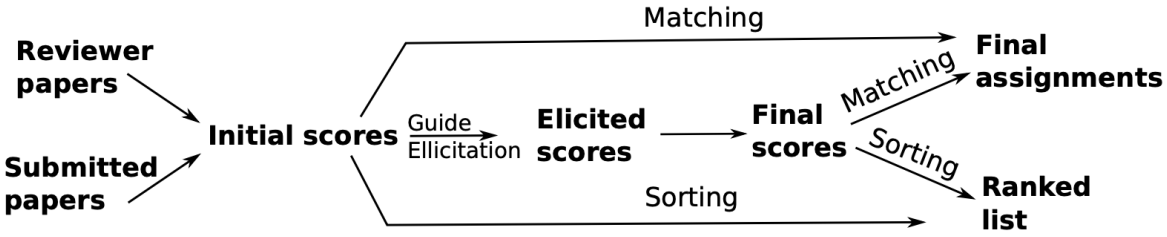



**Figure 0.1.** Typical steps involved in an automated reviewer-submission paper assignment system for a conference. Notice the initial scores, which are prepared by the automated scoring model, and the elicited scores which reflect the reviewer bids, eliciting their interest and self attested expertise, on a submission paper. Image taken from Charlin and Zemel [1]

High-quality reviews are only possible if the allocated reviewers are the best possible experts from the available pool of reviewers. The problem we are trying to solve is determining expertise.

Modelling expertise requires Natural Language Processing (NLP) because most documents/papers are text based. Automating expertise detection can become challenging for NLP models, if the expert has written multiple papers in different areas of expertise. Identifying what would be the best use of the abilities of this aforementioned reviewer, globally across the conference is essential. We make two key observations in our problem setup:

- Reviewer/submission-paper modelling can be done before the conference.
- Bid predictions can be used if bids are available to evaluate the language models on a downstream task.

The next section highlights our contributions based on the above two points.

It is important to highlight that the scope of this work is restricted to the reviewer and submission paper scoring/pairing, and it does not discuss about paper acceptance or rejections or reviewer feed-backs. We also have not studied in detail reviewer assignment under constraints, aka the matching problem, and shall leave it for future work.

# Contributions

The thesis contributions are outlined as follows:

- We seek to model reviewer self attested expertise on a submission paper. We make use of data collected from previous conferences, namely reviewer and submission paper corpus, along with elicited reviewer bids.
- We explore self-supervised and un-supervised approaches for fine tuning transformers in general and BERT transformers in particular, on the text corpus of a conference.

Due to the nature of conference deadlines, bids from reviewers may not be present as a supervised signal for fine tuning transformers, and hence we developed an approach independent of the bid signal. This self-supervised approach also saves us time, since program chairs of conferences can not be expected to fine tune a BERT model, during their tight peer-reviewing schedules. Instead we imagine a scenario where an unsupervised language model is readily available to use with a downstream model for bid/score prediction task.

- We devise a model to combine the expertise from different papers of a reviewer conditioned on a submission paper, instead of doing simple mean and average representations of individual paper representations. Reviewers, especially senior reviewers may have published hundreds of papers, in diverse areas, over a long period of time. An average representation, as is often used in NLP document embedding approach, would end up losing useful information.
- We have experimented and reported results on two very different conferences, NeurIPS 2019 and PLDI 2021. The results are an interesting exploration of benefits of combining the fine tuning of transformer with reviewer expertise combination models. All results demonstrate their ability to predict a reviewer's interest/ability on a submission paper, with elicited bids used as supervised signals for this downstream bid prediction task.

## Structure of this thesis

To aid the reader, we have broken down this thesis into chapters. Here is an outline and short description of the ideas discussed in each of them:

- Chapter 1 discusses background material necessary for understanding our research and to put into a broader context of where our research stands among others. We begin with a description of the Toronto Paper Matching System (TPMS Charlin and Zemel [1]) which is an existing paper reviewer matching system, used in some AI/ML conferences. We had access to data sets from different conferences collected by TPMS. Next, we do a detailed literature review of all related work in three key areas close to this work in scope a) automating peer-review for conferences, b) expertise modeling, and c) language modeling as a tool for ranking/relevance problems, using text as input. We also cover the latest research and tools in Natural Language Processing (NLP) namely attention, transformers and BERT model.
- Chapter 2 is divided into two main sections. The first section of Chapter 2 talks about different approaches to modeling text into vector representations. We then shift the focus to discuss the different self-supervised approaches, and how self-supervision as

a learning framework is an interesting tool for fine tuning transformer based models. We propose approaches for fine-tuning transformers, which could maximize their semantic information encoding capability. We refer to these approaches as *augmentations* during the fine-tuning task. We then discuss how we developed a model for combining representations from multiple papers written by a reviewer.

The latter sections of Chapter 2 focus more on the results we achieved for both NeurIPS 2019 and PLDI 2021 datasets. We first begin with ablation studies for each of the components of our proposed reviewer expertise combination model. Then we discuss in some detail about the differences in performance between different BERT fine-tuning approaches, combined with different expertise modelling models.

- Chapter 3 summarizes our findings and suggests possible future works.

# Chapter 1

# Background

Section 1.1 till Section 1.2 discuss the various aspects of conference peer-reviewing. Sections 1.3 till Section 1.3.2 introduces the TPMS system. Beginning from Section 1.4.3 we discuss different approaches to preparing representations of text, and their progress from simple term frequency models to state of the art BERT transformers.

## 1.1. Peer-review in Academic Conferences

Research in computer science is fast, particularly in machine learning and adjacent fields, and gets disseminated quicker than in other fields of science. Ideas are adopted and improved on a much faster scale. As a result, conferences have been more impactful than journals in the field [2].

Peer-reviews can be single-blind or double-blind. Under a single-blind review system [3], anonymous reviewers know the authors of a paper and their affiliations. A double-blind review system is one, in which neither the reviewers can establish the paper author's identity, nor the authors of the papers are aware of their reviewer's identity. The Mathew effect (Merton [4]) highlighted the risk of greater acknowledgment of contributions by scientists of eminent standing, compared to authors who are less well known. To avoid this problem, presently most conferences in ML/AI adopt a double-blind review system. [4]

Academic conferences have a team of organizers who take care of the process. Program chairs are senior members of the research community who are responsible for selecting a program committee and overseeing the conference proceedings and scientific program. A program committee is a group of established researchers who are responsible for assigning reviewers to the submissions, evaluating the quality of work, and deciding for acceptance or rejection based on reviewer feedback. The program committee is also responsible for tackling the ethical issues arising from plagiarism, simultaneous submissions, or conflicts of interest

[5]. Large conferences such as NeurIPS now have an additional hierarchy of senior area chairs and area chairs.[1]

The assignment of reviewers by an area chair, assumes that this area chair is aware of the expertise of the reviewer, and their suitability for a newly submitted paper. As we would expect, this is not always humanly possible due to the scale of the problem, coupled with tight timelines. Sub-par reviewer assignments can lead to disappointment from the authors of the submitted papers.

## 1.2. Overview of the Bid prediction process

Tools, such as Microsoft Conference Management Toolkit (CMT) can begin with a corpus of papers published by a reviewer.[2] These pre-collected reviewer papers can act as rich sources of information for modelling reviewer expertise by an automation tool.

To ensure that there is a way to cross-check our model predictions on expertise, a conference tool can elicit bids on a submission paper asking for reviewers self-attested expertise and interest. In our approach, we are aware that there is no ground truth of a) reviewer expertise b) or if a reviewer's current interests match their actual expertise. Hence we have used 'Bids', evoked as part of the current Toronto Paper Matching System (TPMS) [1] as proxies, for the reviewer's self-attested expertise and current area of interest. Note that the current TPMS does not use bids for the reviewer score prediction, and just uses it for the assignment under constraints.

We assume that most conferences would be allowing reviewers to view the abstracts, and form a bid decision based on their interests. The main motivation [1] for automating the reviewer assignment process is to reduce the time required to assign submitted papers to suitable reviewers and (ideally) improve assignment quality.

Before going any further, we present here an overview for the steps involved in predicting relevance of reviewers for a submission paper. This reviewer submission paper pairing based on relevance is often referred to as the scoring or bid prediction problem in our work. Figure 1.1 shows the bid prediction process we follow.

We start with the set of papers $p_1, p_2, \ldots p_i$ which are submitted to a conference for peer-review. Each of these papers $p_i$ needs the most suitable reviewer $r_j$. These submission papers are changed to vector paper representations $v_{p_i}$, which can be used as an input for a machine learning model. For each of the reviewers who have registered to review for the conference, we have a corpus of their previously published papers, shown here (Figure 1.1) as $p_1, p_2, \ldots p_k$. We convert each of these reviewer papers to their vector representations, and learn to aggregate/combine them to a vector representation $v_{r_j}$. These vector representations

[1] Area chair guidelines NeurIPS 2017 https://nips.cc/Conferences/2017/PaperInformation/SeniorAreaChairInstructions
[2] CMT homepage https://cmt3.research.microsoft.com/About

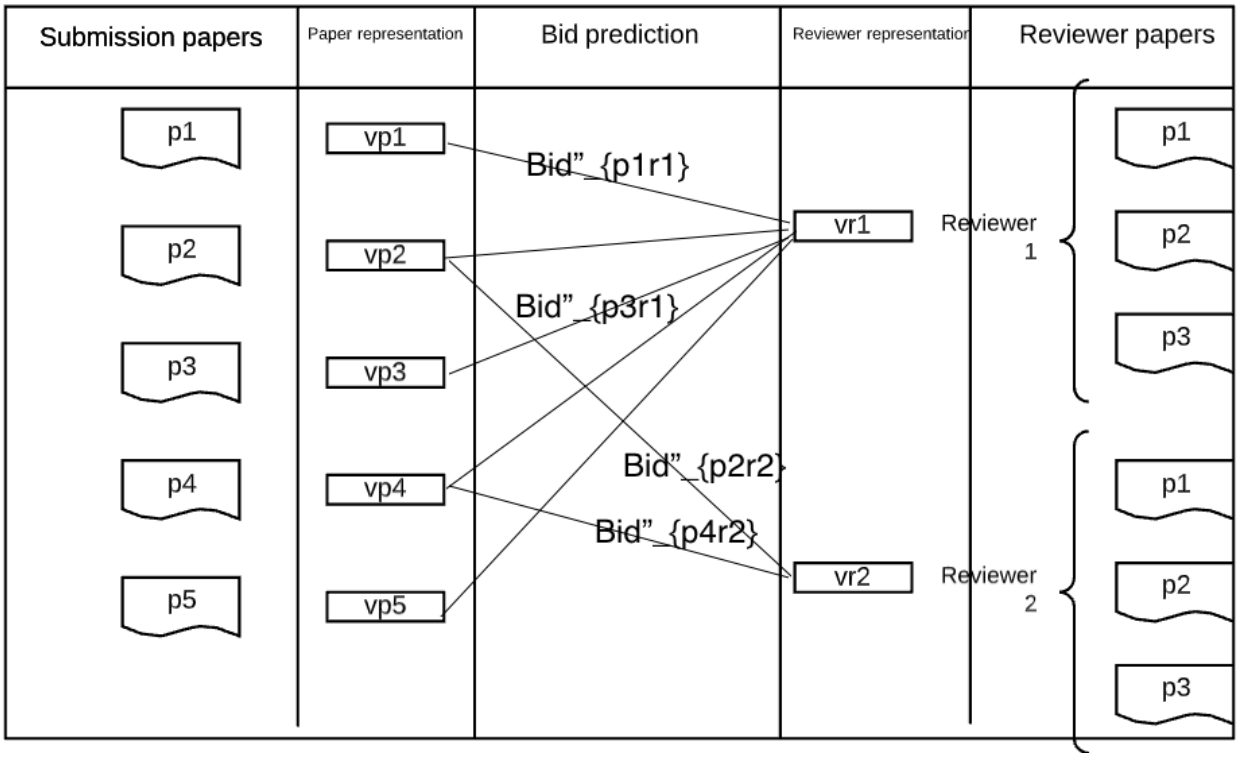



**Figure 1.1.** Overview of the process followed by us to develop: 1) paper representations for submissions and reviewers 2) combining reviewer expertise from the set of papers written by a reviewer 3) predicting the relevance of a reviewer to a submission paper, and the bid which this reviewer would make to show their interest.

are assumed to reflect the reviewer's expertise and we use these representations $v_{p_i}$ and $v_{r_j}$ to predict their $Bid"_{p_i r_j}$. Each $Bid"_{p_i r_j}$ is an anticipation of what the reviewer would bid upon seeing a submission paper, based on their expertise and interest. Effectively these bid predictions represent the relevance of a paper $p_1, p_2 \dots p_i$ for a reviewer $r_1, r_2 \dots r_j$ and indicate a degree of reviewer's expertise and interest in that topic area.

## 1.3. Exisiting system in place: TPMS

TPMS is the Toronto Paper Matching System (Charlin and Zemel [1], Charlin et al. [6]). It was first developed in 2009 [7]. Since then TPMS has been used in some AI/ML conferences, most notably all NIPS/NeurIPS from 2009 to 2020, and ICML, CVPR, ICCV over the same time frame. TPMS tackles this problem by predicting the suitability of reviewers to conference submission papers.

Unlike an area chair, an automated tool can be aware of all the topics involved in an individual submitted paper, and make a decision based on the expertise areas of all reviewers present in the pool. Automating this reviewer assignment process not only decreases the time but can also be seen as a way to make the best global use of all available resources.

Before the assignment of reviewers can start, reviewers are required to complete their research profiles on TPMS/CMT. Early automation tools asked for research keywords (subject areas) to complete user profiles, but TPMS can make use of the publications of reviewers directly. For each conference, TPMS has access to a corpus of publications from reviewers who have registered, and the abstract text and complete PDF of submissions.

TPMS then tries to learn a similarity score between a reviewer-submission paper pair, to predict reviewer suitabilities. Datasets were captured for different conferences and we were fortunate to have access to the datasets for NeurIPS 2019 and PLDI 2021. [3](Programming Language Design and Implementation)

Once reviewer profiles are complete, bids can be evoked from reviewers on submission papers to get an idea of the self-attested suitability/interest of a reviewer-submission pair. It is important to note that these self-attested bids for expertise can often be noisy. Bids as self-assessment can have some obvious shortcomings. Other possible sources of noise in the dataset can be due to:

- Faulty bidding by reviewers due to a wrong click. Or some reviewers could be filling not interested bids, without even going through the submission abstracts.
- Lack of a reviewer's willingness to continue working on an area where they may be already established. Bids are a conciliatory step, to be used as signals of interest. Without bids in the process loop, area chairs would be forcing experts to work based on their past abilities. Some reviewers could be minimizing their chances of getting papers where they have actually expertise, but do not work on it anymore.
- Absence of a consistent understanding of bid values and their meanings. One reviewer might be ready to bid 100 (on a scale of 0 to 100) to signal their interest, while another might be more conservative in their bid.

At this moment, we would like to point out to the reader that TPMS does not use bids for it's reviewer suitability/interest prediction process. It only uses bids at a later stage for reviewer assignment under constraints.

## 1.3.1. Modeling of research expertise

**1.3.1.1. Vector representation of scientific publications.** TPMS first develops a vector representation of the reviewers and the submission papers, which we shall often refer to as paper representations. It then uses these representations as inputs, to find a similarity score between a reviewer and submission paper pair , $(p_i, r_j, s_{r_j,p_i})$. The key assumption is that a reviewer's written work reflects their area of expertise.

Parsing the text of research papers requires them to be tokenized into individual units. Tokenization is the breaking of a text sequence into individual words. These tokenized words

[3]https://conf.researchr.org/home/pldi-2021

can then be used to build a vocabulary of the words present in corpus. Sometimes a word would not be seen in the corpus at training time, but be present at test time.

There are two existing approaches in TPMS for preparing a paper representation:

- Bag of words based term frequency representations (BOW)
- LDA based topic representations

**1.3.1.2. Bag of words approach (BOW).** Bag of words approach, often also referred to as BOW, creates a vector representation of the papers. It encodes papers using the frequency of each of the words/tokens present in that paper. Stop words removal is an essential step for BOW to limit the vocabulary size, and to make sure only the important words from the corpus are learned. It is important to note that while preparing our vocabulary for BOW, TPMS uses only the submission papers. The reason behind it, is that submission papers represent the new research and evolving vocabularies, and hence it is essential to be able to model them, instead of the reviewer corpus which could be wider in focus than the conference, and maybe out-dated at times.

BOW is a simplified case of a Language Model. It sees all words as independent of each other. More generally, Language Models (LM) learn to predict the probability of a word 'w', given its previous word/context, 'c', $P(w \mid c)$. This is unlike BOW which does not have a way to encode context from neighbouring words.

Charlin et al. [6] use a multinomial, $P(w \mid d)$, over words w, observed in a document d. The maximum likelihood estimate of $P(w \mid d)$ is the word count divided by total number of words in the document

While using LM we have to be aware that not all words present in a submission paper corpus would be present at test time for reviewer papers. Since we have chosen to build vocabulary from the corpus of submission papers, there are some smoothing techniques for LM to avoid problems caused by Out of Vocabulary (OOV). For example, TPMS uses Dirichlet smoothing ( Equation 1.3.1 ) to solve for OOV problem [6].

$$P(w \mid d) = \frac{T_d}{T_d + \mu} P_{mle}(w \mid d) + \frac{\mu}{T_d + \mu} P(w). \tag{1.3.1}$$

Where $P(w \mid d)$ models the probability of a word $w$, conditioned on the document $d$. $P_{mle}(w \mid d)$ is a maximum likelihood estimate of the words distribution based on word frequency count. $P(w)$ represents probability for word $w$ in the whole corpus. $T_d$ is the count of words in a document $d$. $\mu$ is a smoothing factor, which can be seen as adding pseudo counts in the probability distribution for unseen words.

**1.3.1.3. LDA approach.** LDA stands for Latent Dirichlet allocation (Blei et al. [8]). It is an unsupervised ML model, used to learn topics present in a corpus of text documents. A topic is a distribution of words. Topics are learned from word co-occurence patterns. These learned topics can be used to create vector representations for submission and reviewer

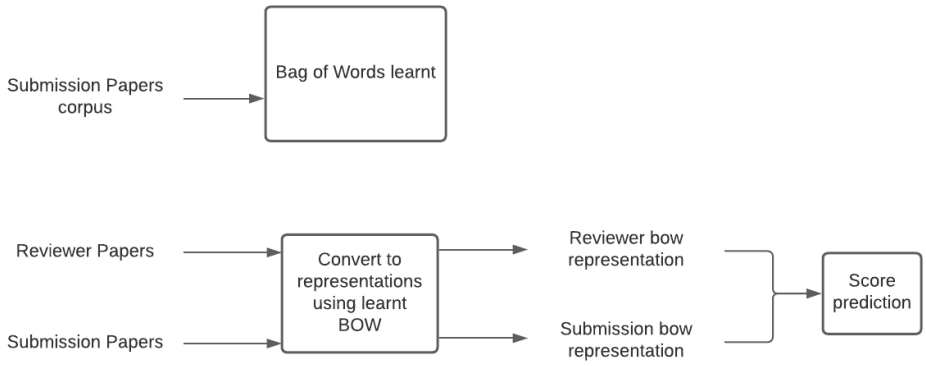



**Figure 1.2.** Work flow for reviewer bid prediction using BOW representations. The BOW vocabulary is developed using only the corpus of submission papers. Each of the papers have been converted to BOW paper representations, and then used as input for the bid/score prediction task

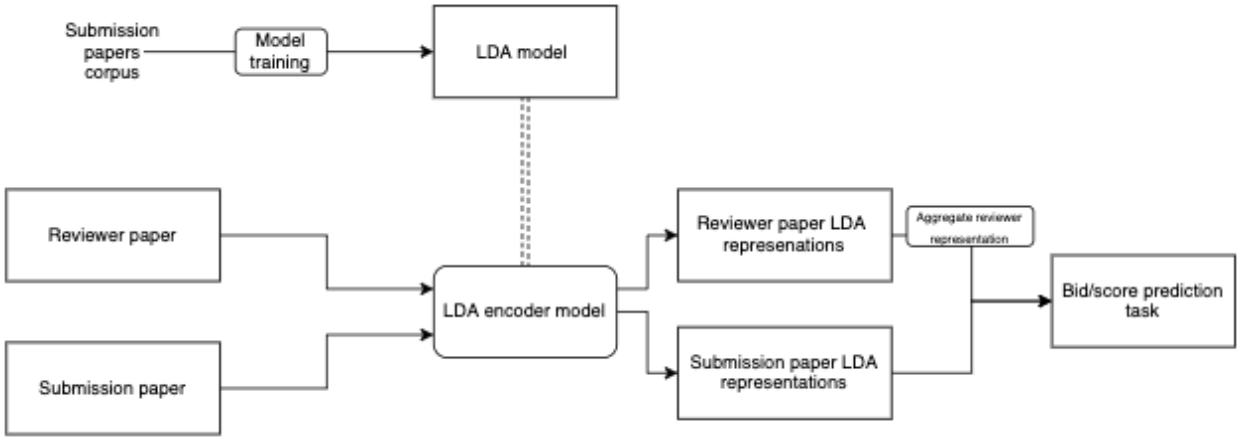



**Figure 1.3.** Work flow for bid prediction using LDA representations. Note that here too the LDA vocabulary, and model is developed using only the corpus of submission papers.

documents using the topic probability values inferred by a trained LDA model. TPMS [1] utilizes the topic proportions as found by LDA to represent documents.

In the next section (1.3.1.4) we explain how we can use bids as our supervised signals. Bids from reviewers act as data labels which we can use with our paper representations (BOW or LDA). We are aware that these bids can be noisy because many reviewers could manipulate the system, but this is the best-supervised signal we have got at the moment.

**1.3.1.4. Bid prediction as a supervised learning problem.** Once we have the paper representation of both reviewer and submission papers, we can use them as inputs for a

prediction tasks. As discussed before, TPMS evokes bids from reviewers as part of its workflow. These bids can then be used as labels for a supervised score/bid/suitability prediction model. In its simplest form, this scoring can be done using a regression model:

$$s_{r_j p_i} = \theta_{r_j}^{\top} f(p_i)' \tag{1.3.2}$$

The equation 1.3.2 models the score, $s_{r_j p_i}$, of suitability between a reviewer $r_j$ and submission paper $p_i$. $p_i$ can be some form of representation of the submission paper text, and $\theta_{r_j}$ is a parameter modelling the reviewer expertise.

There is also a modified version, formulating the scoring between reviewer and submission paper pair (Equation 1.3.3):

$$s_{r_j p_i} = b + b_{r_j} + (\theta + \theta_{p_i}) f(p_i)' + (\omega + \omega_{r_j}) g(A_{r_j})' \tag{1.3.3}$$

Equation 1.3.3 has parameters for each of the individual papers ($\theta_{p_i}$) and reviewers ($\omega_{r_j}$). There are also global parameters $\theta$ for all papers, and $\omega$ for all reviewers. Parameters $b$ and $b_{r_j}$ model the biases involved in this setup. This form of global and individual parameter modelling was proposed by John Langford [1].

The score $s_{r_j p_i}$ prediction task can be modelled as regression, and trained using the Mean Squared Error (MSE) objective, between predicted bids and actual bids received from the reviewers.

There is another approach proposed by the authors, BPMF (Bayesian probabilistic matrix factorization) [6] which uses only evoked bids/preferences but no content information from the paper. But we will not go into its details in this work as it is tangential to our work.

### 1.3.2. The matching step

The matching problem between reviewers and submission papers, once reviewer-submission paper suitabilities are present, can be expressed as a Mixed-Integer Linear Program (MILP) with constraints. Since the most suitable solution for a single reviewer submission-paper pair may not be a globally best solution, hence the MILP formulation is a natural solution. After calculating the similarity scores, TPMS solves a MILP with the following constraints

- Each paper must be reviewed by a given number of reviewers, $R_{number}$
- Each reviewer should not get assigned more than $P_{max}$ papers.
- Each reviewer must have a minimum load $P_{min}$.

The resulting MILP is

$$\max_{y} \sum_{r_j} \sum_{p_i} s_{r_j p_i} y_{r_j p_i} \tag{1.3.4}$$

subject to

$$y_{r_j p_i} \in 0{,}1, \forall r_j, p_i$$

$$\sum_{r_j} y_{r_j p_i} = R_{target}, \forall p_i$$

$$\sum_{p_i} y_{r_j p_i} \leq P_{max}, \forall r_j$$

$$\sum_{p_i} y_{r_j p_i} \geq P_{min}, \forall r_j$$

Each $y_{r_j p_i}$ denotes an assignment if the value is 1, otherwise non-assignment is 0.

The MILP maximizes the assigned scores while ensuring that the constraints are met.

Stelmakh et al. [9] introduced a minimum fairness notion in their assignment objective. Each set of reviewers assigned to a submission paper, must have sum of expertise/similarity scores above a given threshold.

In the scope of this present work, we will limit the discussion of the reviewer assignment step, and the mixed integer problems. We will not discuss it further, as we are more interested in developing better paper representations of research publications, and their use for bid prediction.

In the next section, Section 1.4, we present other works which are related to automating peer-review in conferences. There are three broad sub-sections namely, automation for reviewer interest prediction, expertise modelling through reviewer key words and published content, and NLP tools for paper representation.

## 1.4. Literature review and related work

### 1.4.1. Automating conference peer-reviewers selection:

Rigaux [10] treat the problem of assigning reviewers to desired papers, as one of predicting reviewer preferences based on a collaborative filtering framework, similar to a recommendation system. Reviewers are not required to rate all papers, which can be a very tedious task. Instead, the model performs the task of filling missing ratings for each reviewer and submission paper pair.

Conry et al. [11] also, adopt a recommender systems-based approach. Multiple sources of information are utilized apart from the reviewer bids data. Parameters for weighing different sources namely abstract similarity between different submission papers, reviewer to reviewer similarities based on co-authored papers, and conflicts of interest are part of their trained model.

Rodriguez and Bollen [12] exploit co-authorship network data to assign suitable reviewers for new submission papers. Each node represents a published paper, and edges represent a

collaborated work. Their approach gives a submission paper-specific weight for each individual represented in the co-authorship network. This approach does not start with a closed pool of reviewers to choose from, and thus can be used to suggest a set of initial reviewers from an open pool of possible reviewers.

Mimno and McCallum [13] aim to model the expertise of a reviewer with respect to the topics of a given paper by finding the affinity of a reviewer-submission paper pair $(r_j, p_i)$. Each reviewer is treated as having expertise in multiple topic areas, represented by the topic distributions in their different published documents.

Balog et al. [14] proposed going through an organization's document repositories for finding experts. We find this approach similar in principle to the one adopted by TPMS [1]. But the major exception is that Balog et al. [14] does not start with clear attribution of documents to their authors, whereas our TPMS [1] dataset has research papers with names of authors clearly mentioned. Hence a lot of their (Balog et al. [14]) focus is on how to attribute a document to an author based on keywords overlap.

### 1.4.2. Expertise modeling

Determining the topics or areas of expertise of a researcher is important for automated reviewer assignment systems. Differentiating new experts from experienced experts, and the expertise level of each researcher can be helpful if we want to assign more challenging papers to more experienced reviewers.

Early approaches to expertise modeling were based on knowledge databases of the experts. These databases could be filled through manual entry of expertise data for each of the experts. Manual data entry is used in skill inventory systems like Skillview, and the SAGE People Finder, used in the human resource management domain.[4] [5] Yimam-Seid and Kobsa [15] queried users about their own expertise. Obviously, these systems would assume that users are honest about their own abilities, and can correctly judge their expertise areas if any, without any intention of malice. Moreover, Balog et al. [14],[16] observed that static databases can become incomplete and antiquated very often. There could also be non-overlap [16] in the finer-level query being used to search for experts and the generic descriptions provided by the users for their expertise.

Probabilistic language modeling techniques [14] have been applied previously in Information Retrieval tasks. A model could rank candidates, based on the probability of the candidate being an expert on the given topic. Balog et al. [14] demonstrated two general strategies for expert searching conditioned on a given document collection. The first approach models an expert's knowledge based on the documents that they are associated with. A textual representation of each candidate reviewer's topic expertise is built. The second

[4]Skillview http://www.skillview.com/

[5]SAGE People Finder http://sage.fiu.edu/Mega-Source.htm

approach instead, locates documents on the topic as a first step and then finds the associated expert. Balog et al. [14] formulate $P(candidate \mid query)$, to find top $k$ candidates as the most suitable experts. It is important to note that these associations could be built based not only on explicitly mentioned authorship but also if the name or email of a candidate appears inside a document text. This is unlike our case, where we shall only consider publications explicitly mentioned as written by authors (or co-authors).

Han et al. [17] proposed a model where expertise is embedded in a vector space. They assume that a task might need multiple areas of expertise, and thus use collaborative networks to route each task through a set of area specialists, till the task is completed. A task can be assigned to an expert if their representation is close to the task embedding. The authors also attempt to model proficiency, where experts must have proficiency greater than or equal to the proficiency required by the task itself.

Qian et al. [18] model reviewers with the same social group, position, and affiliation as having similar research preferences. This is formalized through a weakly supervised- factor graph. h-index of authors, number of publications and citations, are used to model the correlation factor function between researchers.

In all these expertise modeling paradigms, one very important point to always consider [13] is the assumption that each author of a paper has expertise over all the topics of a paper, which may not be always true. Often people are invited to co-author a paper, on very specific sub-sections.

### 1.4.3. Background in NLP

A reviewer's previous publications can be a useful source for determining expertise[1]. The different approaches on expertise modelling in section 1.4.2 motivated us to explore content based expertise models. These models would aim to represent text data as vector representations.

Term frequency (Robertson and Zaragoza [19]) based models are very convenient for exploitation, as even in their most basic form, without any relevance labels, they deliver results in information retrieval. Though using term frequency models assumes all documents are talking about the same topic. Using LDA (Blei et al. [8]) we could assume that each word incidence [19] is associated with a topic probabilistically, and in turn a given document contains a number of topics. Hence a topic model (LDA based) would be more efficient at capturing incidence of different topics within a single document.

Another approach, called a Language Model (LM) can be defined at its core as a probability distribution over the words in a text (sentence, document), based on the preceding context. Mathematically such a probability distribution can be expressed as [20]:

$$p_\theta(w_t \mid c_{t-1}) = p_\theta(w_t \mid w_0, w_1, ...., w_{t-1}), \forall t \leq T \tag{1.4.1}$$

Parameters $\theta$ model the word distribution $p_\theta(w_t)$ for a given portion of continuous text, where the length of the whole text sequence is $T$.

LM [21; 22] can also be applied for information retrieval, and document ranking problems, with the assumption that a document can be a match for a search query if and only if the document model is likely to generate the query. A probabilistic relevance model would want to develop a ranking function, which can rank documents according to their relevance, conditioned on a query. The applications could be search engines. In its simplest form, it can be modelled as:

$$sim(d_j,q) = \frac{P(R \mid \overline{d_j})}{P(\bar{R} \mid \overline{d_j})}V \tag{1.4.2}$$

Where document $d_j$ is investigated for the probability of relevance with a query $q$. $R$ would be the set of documents labeled as relevant by a set of users for query $q$, while $\bar{R}$ would be a set of documents seen as non-relevant. Practically this would mean, more the number of words matching in a document with a query, the greater the chances of a relevance match, and higher similarity score.

BM25 (Robertson and Zaragoza [19]), was a pioneering approach for information retrieval using language modeling. BM25 is abbreviation for Best Matching 25. The score used term frequency count of matching words in a document $d$, with the query $q$ and then weighted them with an Inverse Document Frequency (IDF) term which gave weightage to terms that were rarer in the corpus of texts.

The advent of neural network based models has changed the field of natural language processing. Mikolov et al. [23] introduced the Skip-gram approach (known popularly as Word2Vec), to learn continuous vector representations of words, using an unsupervised approach. Skip-gram begins with an un-labelled text corpus, where it tries to predict the context words based on a target input word. Effectively it converts original unsupervised problem to a pseudo supervised setting. Once the model is trained, the pre-trained embeddings from this model can be leveraged for later downstream tasks. Le and Mikolov [24], in their important work, pointed out that only using bag-of-words features dismisses word ordering and does not preserve word semantics. They proposed an unsupervised algorithm that learned fixed-sized vector representations from texts of varying sizes such as sentences or documents.

Developing sentence or paragraph level representation from word representations is not that trivial. Liu et al. [25] show that sentence-level embeddings can be approximated using some heuristic to combine the vector embeddings of the individual words in a sentence. The compositions could be additive or mean or max-based approaches. The sentence-level embeddings between the candidate and target response can then be compared using different distance measures. Dai et al. [26] showed the usefulness of paragraph vector embedding approaches by finding related articles on ArXiv.

Doc2VecC (Chen [27]) proposed representing documents as the average of word embeddings of all the constituent words in a document during the learning process, instead of post-processing unlike the previous approaches. Some words were removed at random during training time, to add noise, and act as model regularization. They added a label id at chosen levels of text/document granularity, which could be either a sentence, a paragraph or the whole document, with Word2Vec embedding inputs.

Having seen the progress in other NLP tasks, we are aware that TPMS might benefit from using better paper representations that can find out what each of the papers are talking about. Capturing the semantic meaning behind each of the research publications would help us avoid developing an over-fit solution, which could become useless for other word distributions, such as different conferences. Words occurrences themselves may be misleading, because many people working in the same research area could be using different words to imply the same research. For example some people might prefer the term 'sequence models' over 'recurrent neural networks' in their usage.

**1.4.3.1. What is a downstream task in NLP?.** There has been a trend of using pre-trained models with different tasks. Such tasks that a model has not been specifically trained for, and could be very different from the pre-training tasks seen while training language models are generically referred to as downstream tasks. Within an NLP framework, pre-training tasks aim to build as much semantic understanding as possible. These pre-trained models could then be used on downstream tasks directly or with further fine-tuning, such as Sentiment analysis, Named Entity Recognition, Machine translation, or Summarization.

Current state of the art in NLP is dominated by transformer based pre-trained models. Hence it is important for us to discuss some of their basics. Since most of the transformer based models use attention and self-attention as the fundamental units, we start with some background on them.

**1.4.3.2. Attention.** Vaswani et al. [28] proposed a general definition of attention. They see attention as mapping a query and a set of key-value pairs to an output. The query $Q$, key $K$, and value $V$ are learned using weight matrices $W_q, W_k, W_v$. The dimensions of these weight matrices are all the same, of dimension $d_k$, $[pad, pad]$, where $pad$ is some fixed length padding value.

The final output of attention module is a weighted sum of these matrix values, giving us 'scores' for each of the values conditioned on our query.

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{1.4.3}$$

The term $\sqrt{d_k}$ is added for stability. Figure 1.4 shows the flow of information for dot product attention as introduced by Vaswani et al. [28]. Attention-based models have the added

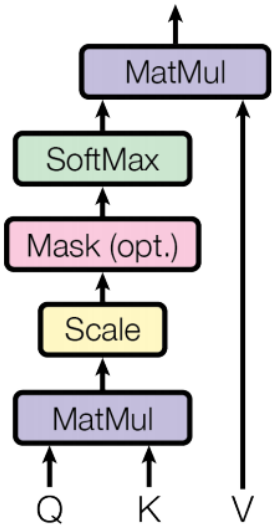



**Figure 1.4.** Scaled dot product attention. Attention has been simplified to a set of weight matrix multiplications. Image taken from Vaswani et al. [28]

advantage of being more parallelizable which means better usage of GPUs, and significantly less training time requirement.

A Multi-head attention is a combination of attention units, where the number of heads is the number of such repeated units. There are multiple sets of $W_q, W_k, W_v$ matrices, each of which learns its own approach to attention scoring. For our model, having a multi-head attention model did not make any significant difference to our results. Figure 1.5 shows components of multi-headed attention.

**1.4.3.3. Self-attention.** Attention was first developed (Bahdanau et al. [29]) to align output of one component, with the input of another. For the case of machine translation, Bahdanau et al. [29] aligned encoder outputs, with the decoder inputs. In this scenario (Equation 1.4.4), attention ($\alpha_{ij}$) was a component distinct from the encoder and decoder, which learnt a context vector $c_i$, based on weighted sum of previous encoder states ($h_j$):

$$c_i = \sum_{j=1}^{T} \alpha_{ij} h_j \tag{1.4.4}$$

Self-attention, removes this separation of encoder and decoder. In context of NLP, this could mean that every input element of a sequence, is attended to by other elements in the same sequence. This helps the model develop attention scores in both directions.

A Uni-directional model is trained only to learn relations in one direction , usually the way a language is written. Since a uni-directional model would be oblivious to information in a sequence which has not happened yet, they often fail to capture the contextual meaning. Hence they tend to underperform in tasks where context dependent information is critical.

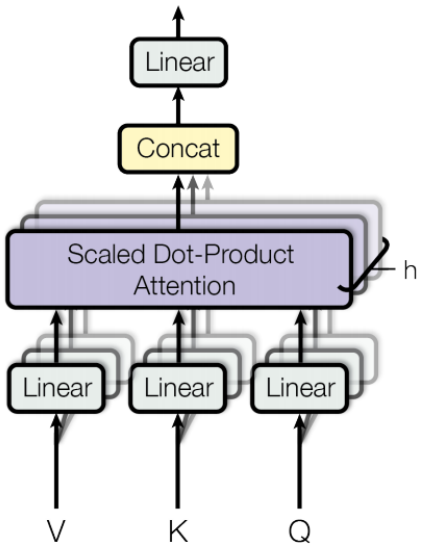



**Figure 1.5.** Multi head attention. There are multiple sets of $W_q, W_k, W_v$ matrices, each of which learns its own approach to attention scoring. Image taken from Vaswani et al. [28]

For example for a sentiment analysis task, the phrase "silver is empty in the bank" has negative connotations, and should be identified so. While the phrase "silver is empty in the banks of river ..", could be a neutral expression describing something about a river's geology.

Self-attention when used for a text sequence, learns the relation of each token with the other tokens in a given text input. These different relations could be capturing different syntactic properties of a language. Due to its $O(N^2)$ complexity in operations, self attention makes processing long sequences such as documents, computationally expensive , and limits their use. There has been some research into sparse attention to solve for this problem, but we will not go into it in this work.

**1.4.3.4. Bidirectional Encoder Representations from Transformers (BERT).** BERT (Devlin et al. [30]) stands for Bidirectional Encoder Representations from Transformers. Given the recent success of BERT everywhere, and transformer models in general, BERT seemed like the obvious choice for an embedding model. BERT has been pre-trained on unlabeled texts using semi-supervised approaches. The pre-training tasks are self-supervised approaches, where a huge corpus is used to pre-train the models, without the use of any human annotated labels. For BERT these were the two pre-training tasks:

- Next sentence prediction: Model tries to predict if given two sentences as an input, is the second one following the first one in meaning.
- Masked language modeling: Words are removed from an input sequence at random, and then are to be predicted by the model based on context from neighbouring words.

It is important to note that BERT is pre-trained on the mask language model (MLM) and next sentence prediction (NSP) task simultaneously. At training time Devlin et al. [30]

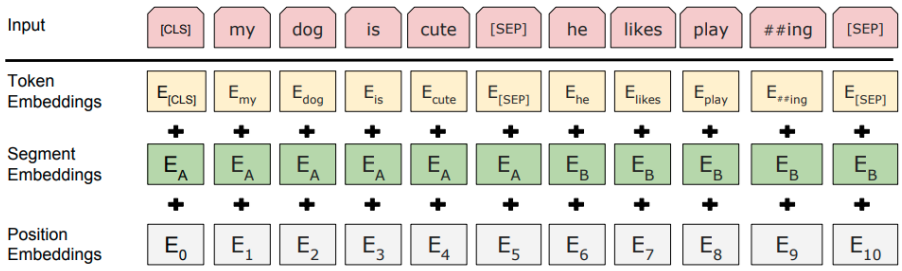



**Figure 1.6.** BERT input representation. Text is tokenized into individual units and combined with position embeddings to get an idea of text order. For our case, we did not need segment embeddings. Image taken from Devlin et al. [30]

added a [CLS] token symbol before every input text. This token on its own did not have any semantic meaning originally, but the nature of the pre-training tasks led it to becoming a summarizer of the information contained in a sentence.

These learnt BERT embeddings can be applied with some additional fine-tuning, to a wide range of tasks. BERT word embeddings change depending on the context of a given word, and thus are sensitive to word order.

**BERT limitations:** BERT has been trained to take less than 512 tokens as input, due to the $O(N^2)$ computational complexity. We can pass the BERT model a whole sentence as an input (if number of tokens $< 512$), and get context dependent representations, unlike other approaches such as Word2Vec, which give static word embeddings.

Some research and heuristics have been developed around building representations for sentences or paragraphs from BERT word representations. The final layer representation of [CLS] token is a fixed sized vector. Usually for sequence classification tasks, it is assumed to be equivalent to a pooled/ summarized representation of the input sequence. This vector of size $'768'$ dimensions can be used for downstream tasks. BERT authors, Devlin et al. [30], used the final hidden layer representation of [CLS] token, as aggregate representation for a sentence, for fine tuning on GLUE tasks.

Instead of using [CLS] tokens as summaries, there have been some other approaches based on averaging of individual word embeddings. Though intuitively it seems a simple averaging of word embeddings would lose all the context dependent nuances of the BERT word embeddings, rendering it futile.

SciBERT [31] is a BERT based model, which has been specifically trained on scientific domain data. Due to its pre training on a large multi-domain corpus, we realised it is not that useful for CS/ML/AI conferences due to the different semantic meanings of words across different scientific fields.

RoBERTa [32], XLM [33] are some of the other successful cases from NLP, where models have been pre-trained with a a more traditional self-supervised approach. In the next section,

Section 1.5, we shall discuss how self-supervised learning has evolved from triplet loss based networks to more recent contrastive learning approaches.

An important problem that occurs if we learn word-level embeddings, is how to turn these into sentence or document-level representations. Simple approaches like averaging of word embeddings have been shown to work. Another way to calculate sentence-level embeddings is using vector extrema ( Forgues et al. [34]). For each dimension of the word vectors, take the most extreme value amongst all word vectors in the sentence, and use that value in the sentence-level embedding. Intuitively, this approach prioritizes informative words over common ones; words that appear in similar contexts will be close together in the vector space.

## 1.5. Towards self-supervised learning

Section 1.4.3.4 discussed some recent developments in NLP, most notably the use of pre-trained models for embeddings and text representations. Now that we have already presented background knowledge in NLP, we shift towards the other direction related to our work. This section shall focus on approaches to self-supervised learning.

Pre-trained models in NLP often need some kind of further training on the chosen data sets. The training can be easily done within a supervised framework if the training labels are available for the task, but that is not always the case. The progress in supervised learning has been impressive, but it has been dependent on labeled data. Good quality annotations and labels are required as model training is sensitive to noisy labels. For these reasons, data labeling can be a laborious, meticulous, and expensive process.

Unsupervised learning is a class of models that does not require training labels. According to Tschannen et al. [35],these approaches aim to learn a function that embeds data into a lower-dimensional space, which can then be used for downstream tasks.

Self-supervised learning is a framework which obtains labels for a dataset, using the dataset itself. The defining feature of self-supervised learning is that we start the process without access to any labels. Under a self-supervised framework, instead of labels we can extend unsupervised approaches by using the structure present in the input data itself [36; 35]. It can be spatial, say patches from the same image can be assumed to have closer representation than patches from different images. Or temporal information, such as temporally close video frames should be embedded closer than further frames.

Since self-supervision approaches often use distance comparison, we would like to mention a distance based network, which though supervised in nature, is similar in principle. Hoffer and Ailon [36] proposed a triplet distance-based model (refer to Figure 1.7) which would learn representations by doing distance comparisons between samples from the same class, and samples known to be from a different class. Even though the triplet distance based

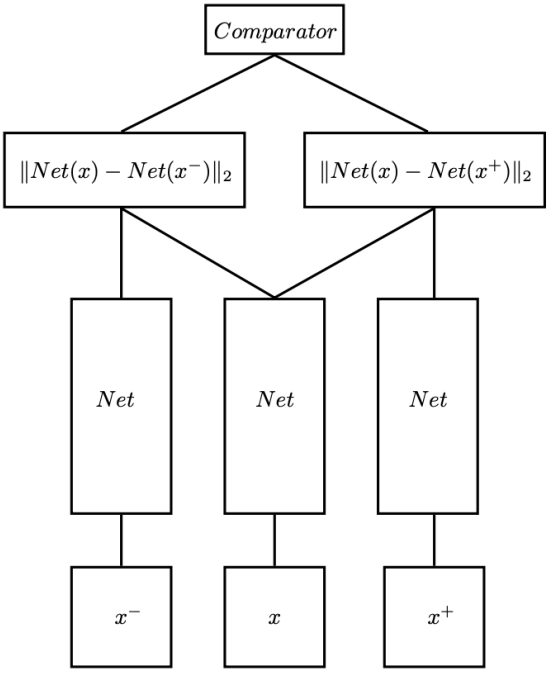



**Figure 1.7.** Triplet loss structure as proposed by Hoffer and Ailon [36]. Samples of base class $x$, negative class $x^-$, and positive class $x^+$ are passed through a network, sharing the same weights, to get representations and then distance compared. Figure is taken from Hoffer and Ailon [36]

network is different from self-supervised approaches, as it has access to class labels, we can notice some similarities in its contrastive loss objective. Samples of base class $x$, negative class $x^-$, and positive class $x^+$ are passed through a network, sharing the same weights. The network then produces three representations, and distance is computed between them. This work was an early inspiration for us to head towards self-supervised approaches. Figure 1.7 shows an example of a triplet distance based architecture.

Recent self-supervised approaches are distinguished by the clever manner in which unlabeled data sets, are changed to a pseudo labeled dataset. Often, these models also make use of contrastive loss objectives to train. A contrastive loss objective brings embeddings/representations of the same sample close in distance, and the representations of unrelated samples far apart in distance (refer to Section 1.5.1.3 for a more detailed explanation).

Within the self-supervised framework, a task agnostic approach (Oord et al. [37]) would be learning representations that are less focused on solving a single supervised task. The features should transfer if the tasks are in reasonably related domains. Oord et al. [37] states that these encoders should encode the underlying shared information in latent space while discarding the local information and noise.

Tschannen et al. [35] pointed out that optimizing the objective at a fine-tuning time for self-supervised learning, say Mutual Information, may not correspond to better downstream performance always. Tschannen et al. [35] posit that there could be a trade-off between the amount of information to be stored by the encoder against how hard it could be to extract it in the downstream tasks.

In the next two sections, Section 1.5.1 and Section 1.5.2, we present two approaches from recent literature for self-supervised learning. We also introduce the reader to their applications in our own encoder training approach.

### 1.5.1. SimCLR: A Simple Framework for Contrastive Learning of Visual Representations

SimCLR was proposed by Chen et al. [38] as a model agnostic framework, for self-supervised learning. The applications shown by the authors were exclusively in Computer Vision, and our work is one of the few ones implementing it for textual data (Jain et al. [39] recently showed a contrastive approach, for learning if software codes are similar in functionality). SimCLR framework uses a 'Contrastive' objective to train an encoder model on unlabeled data. In our implementation, this encoder is the BERT transformer, which we will fine-tune using this SimCLR self-supervised framework.

The authors of SimCLR noted that model performance improved greatly with an increase in the size of batch and found a continuous improvement of performance, with an increase in the number of epochs [38]. The dependence on batch size can be explained by the importance of negative samples, and their comparison or contrast with the positive samples. Figure 1.9 shows a cosine similarity matrix between different positive and negative samples for a contrastive loss objective.

Figure 1.8 demonstrates the SimCLR [38] architecture used by us for fine tuning on submission papers. If there are $N$ samples in the original dataset, the augmentation task creates an extra transformed copy, leading to $2N$ samples in total. The original input is passed through one of the augmentation tasks, as defined in Section 2.2.5. This augmented input is then passed through the encoder and projection layer combination. We also pass the original input, without any augmentation through the encoder and projection layer framework. The encoder is forced to learn to match the two representations since essentially the two inputs are transformations/augmentations of the same original input. In our case, we chose to keep one transformation as identity and the second transformation as one of the text augmentation tasks (refer to Section 2.2.5). We then train our model using the contrastive loss (defined in the Equation 1.5.1), till it goes down to convergence. Once our model has been trained, we shall be using only the encoder and discard the projection layer. Interestingly the SimCLR framework is model agnostic and we can choose any suitable

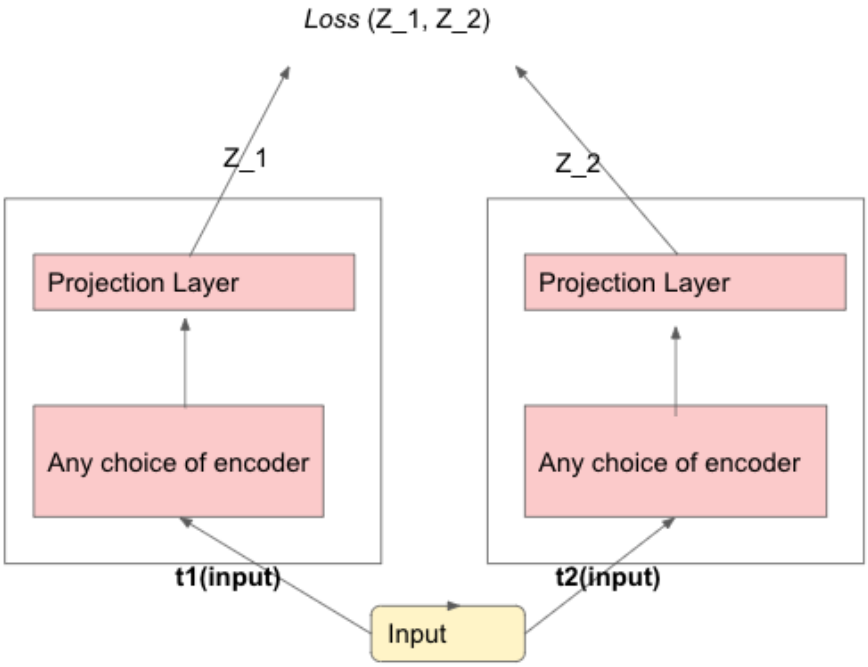



**Figure 1.8.** Architecture for SimCLR fine-tuning. The original input is taken through two transformations and then passed through the encoder and projection layer. The encoder attempts to learn to match the two representations since they are transformations/augmentations of the same original input. In our case, we chose to keep one transformation as identity and the second transformation as one of text processing operations.

neural network, as the encoder for our task. In this work, we have stuck to using a BERT transformer as the encoder. Other authors in self-supervised learning have mostly used 'ResNet' (He et al. [40]) as the encoder (Chen et al. [38]).

The original SimCLR authors Chen et al. [38] demonstrated the use of the SimCLR framework on computer vision data sets. The intuition behind matching the augmented and non-augmented views is that the two views are practically the same thing semantically. For example, flipping an image or rotating it does not semantically change an image, it only superficially alters it. Teaching this task to the encoder inside the SimCLR model forces it to learn the semantic information in more detail. Bachman et al. [41] mention that multiple views of a shared context forces the features to capture information about higher-level factors.

The SimCLR framework involves the use of two neural network modules (see Figure 1.8),

- The encoder
- The projection layer

**1.5.1.1. What is an encoder?** An encoder is a neural network model, which learns to embed the text into a useful latent space. We train this encoder via the Contrastive objective inside the SimCLR or BYOL (check Section 1.5.2) framework. Once trained, we

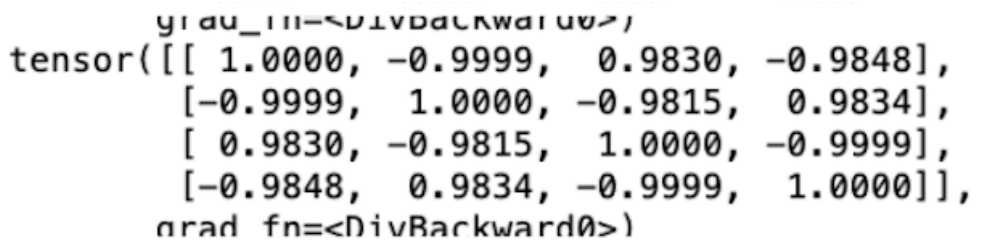



**Figure 1.9.** Inside similarity matrix for a Contrastive loss framework. Similarity values of 1 along the diagonal represent the cosine similarity between positive samples. Similarity values tending towards $-1$ represent the cosine similarity between negative samples.

freeze the parameters of this encoder and can then use it to encode new documents, to get their embedding representations.

**1.5.1.2. What is a projection layer?** Popular approaches for self-supervised learning use a projection layer, trained on top of the encoder during the self-supervised learning task. Once the model is trained, the projection layer is discarded, and only the encoder is used for downstream tasks.

We can create different kinds of tasks by creating different kinds of data augmentations.

**1.5.1.3. SimCLR loss objective.** This is a form of *Contrastive loss*. SimCLR [38] loss is defined as the normalized temperature-scaled cross-entropy loss, or 'NT-Xent' ( see Equation 1.5.1) . We have used this loss objective in our fine tuning regime of BERT transformer:

$$L(z_i,z_j) = -\log \frac{\exp(\frac{z_i z_j}{\tau})}{\sum_{k=1}^{2N} 1_{k\neq 1} \exp(\frac{z_i z_k}{\tau})}. \tag{1.5.1}$$

From the Equation1.5.1, we can observe that the numerator captures the cosine similarity values, $z_i, z_j$ for a pair of samples $i$ and $j$. The denominator is contrasting this with cosine similarity values between negative samples $z_i, z_k$, where $k$ is any negative sample/example for our sample $i$. For a batch size of $N$, we end up with $2N-1$ terms in the denominator, as for every sample $i \in N$ we have produced an augmented sample $j$. Here $\tau$ is a temperature value for the softmax function. We found that the value of $\tau = 0.5$ works well for our model runs. There is some discussion in self-supervised literature about the optimal temperature values, which could be useful for selecting hard negative examples during training, but we did not investigate in that direction.

An intuitive way to debug a contrastive loss-based framework can be to look at the similarity values inside a batch of samples. On observing the similarity matrix (see Figure 1.9), this loss should eventually go to 0s and 1s, to reflect that the model has learned to distinguish between positive and negative samples in the mini-batch.

It was reported by Chen et al. [38] that increasing batch size and dataset led to continued improvement in performance. We observed the same in our case when using larger batch

sizes (12 vs 64 samples in a batch), with a larger data set (NeurIPS 2019 dataset vs PLDI 2021 dataset). This could be due to the presence of a larger number of negative samples in a larger batch size, making it easier for the model to learn to differentiate between similar and dis-similar samples.

### 1.5.2. BYOL: Bootstrap Your Own Latent

BYOL (Bootstrap Your Own Latent) Grill et al. [42] is another approach towards self-supervised learning. Similar to SimCLR, we implemented it for fine-tuning BERT transformer using different augmentation tasks. Figure 1.10 shows two views of the same original input: one is augmented, while the other remains the same. The authors [42] successfully demonstrated that any pre-trained model such as Resnet or VGGnet could be fine-tuned with this framework. The key attraction of BYOL is that this framework does not need negative samples for its training, which gives us more choices for augmentations.

BYOL loss, Equation 1.5.2, is defined as the MSE (Mean Squared Error) between the normalized predictions ($\overline{q_\theta}$) and target projections ($z_\theta$). The normalized prediction is the encoder embedding which tries to match/predict the output of the delayed target projection. The right side term of the loss in Equation1.5.2, measures the inner product between ($z_\theta$) and ($z_\xi$) Both modules share the same weight parameters, but gradient back-propagation only occurs in the prediction module.

$$L_{\theta,\xi} \approx \|\overline{q_\theta}(z_\theta) - z_\xi\|_2^2 = 2 - 2.\frac{< q_\theta(z_\theta, z_\xi^{'}) >}{\|q_\theta(z_\theta\|_2.\|z_\xi^{'}\|_2} \tag{1.5.2}$$

Effectively, Equation 1.5.2 can be seen in the simplified form shown in the Equation 1.5.3. The loss $L_{BYOL}$ would be low when $A$ and $B$ have high cosine similarity, which is what we wanted to optimize the model for.

$$L_{BYOL} \approx 2\Big(1 - .\frac{< A, B >}{\|A\|_2.\|B\|_2}\Big) \tag{1.5.3}$$

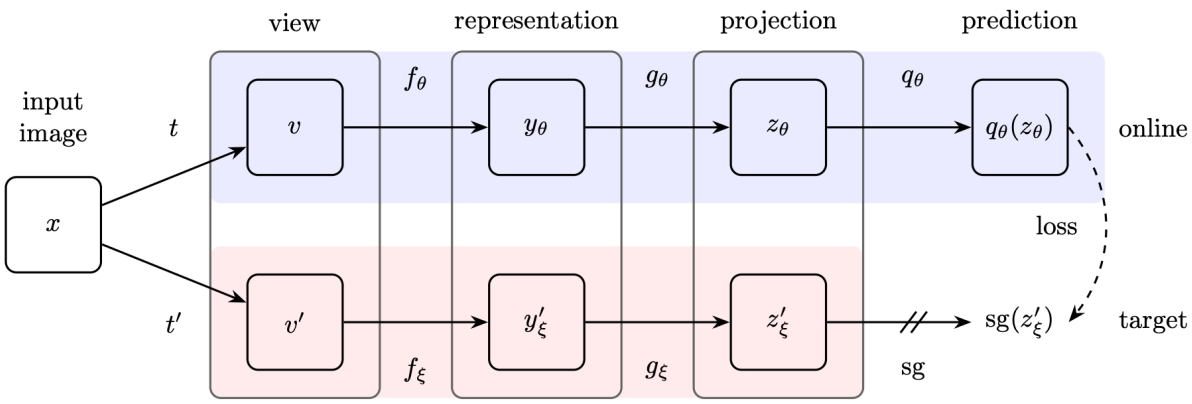



**Figure 1.10.** BYOL network architecture. The original input is taken through two transformations and then passed through the encoder and projection layer. The encoder learns to match the two representations, but in this case, unlike SimCLR there is no contrastive learning and no need for negative samples. Image is taken from Grill et al. [42]

# Chapter 2

# Improving paper representations for peer-reviewing

This chapter shall focus on the contributions of our work. First, we present the challenges present in preparing text representations using traditional BOW and LDA approaches. Then we shift the focus to discuss fine-tuning such generic representations for a specific conference either using the original model or different self-supervised approaches namely SimCLR and BYOL. We propose several augmentations for the fine-tuning task and talk about why we chose to implement them. We then discuss how we developed a model for combining representations from multiple papers written by a reviewer. In the end, we document our experimental results using the different representation and downstream model combinations.

## 2.1. Issues with BOW and LDA embedding approaches

The existing TPMS systems use paper representations built using BOW or LDA models. The performance of the downstream bid/suitability prediction task is tied to the quality of paper representations passed as input at this step. One of the issues with BOW and LDA embeddings is that they can not give context-dependent word or sentence embeddings. Context dependent representations change word representations based on the neighbours of that word. For example 'neural' should have different word representations when used in the phrase 'neural model', against the phrase 'neural psychiatry'. In other words, BOW and LDA model co-occurence, but not word ordering and hence are not the best approaches to model word semantics.

Using n-grams can bring some sense of word order within phrases [24] but even then there can be different sentences with the same choice of words, yet different meanings. Both BOW and n-gram approaches would fail [24] to capture distances between words that could be similar in meaning. Large vocabulary sizes can lead to a very sparse BOW representation for

documents, which can be another problem for technical spaces with many evolving technical terms.

Recent methods including transformer-based approaches (BERT, RoBERTa) try to solve these problems, by projecting word meanings into a continuous embedding space, while taking into account word ordering. The word embeddings of a BERT transformer differ based on the context. The same word could have different latent space embeddings, depending on the words in its neighborhood, giving better flexibility to the paper representations. Hence context sensitive representations could be useful, where terminologies are sensitive to the nuances of sentence construction. BERT has been very effective since its inception in many different text representation tasks. Hence it was a natural choice for us to start building better paper representations using BERT as the starting transformer model.

## 2.2. BERT based representations

The authors of BERT [30] talked about two possible approaches for using pre-trained BERT models for downstream tasks:

- Feature-based approach: Where we extract representation from some of the intermediate hidden layers, and use them on our task, without any gradient propagation for modifying them.
- Fine-tuning based approach: Where we train all or some parameters of the BERT transformer model for our specific task. The gradient is back-propagated through chosen layers (parameters to update) to tune them according to the task. Back-propagating gradients through all the parameters is computationally expensive and not often done with limited compute resources.

### 2.2.1. Fine-tuning based approach

BERT has been pre-trained on a corpus containing the English Wikipedia and books corpora. Due to this, the word distribution present in it could be very different from the nature of words present in a scientific conference.

Training a BERT model from scratch is a computationally expensive and time taking process. Hence we do not attempt to pre-train but only fine-tune the submission papers corpus for a conference. We envision this is how a program chair could use it when deploying this for an actual conference. For example, fine-tuning can prepare an encoder in advance when the participants have submitted their submission papers or abstracts. Then while the area chairs wait for the reviewer bid interests to come in, they can pre-train the models on the submission corpus for the transformer encoder.

To adapt our transformer models to the new evolving terminologies in AI/ML or computer science, we can use a transformer model trained on a very large generic language corpus and

then fine-tune it for our specific conference submission papers corpus. In fact, NeurIPS 2019 had 6,810 submission papers. There were also 3,961 reviewers, each with their corpus of published papers, sometimes containing as many as 100 papers each. A fine tuning-based approach usually depends on the downstream task involved. In our case, we show in Section 2.2.5, how we have leveraged self-supervised learning to do fine tuning without being aware of the downstream bid prediction task.

### 2.2.2. Feature based approach

Feature-based approaches instead of doing a gradient back-propagation through the parameters of BERT model, just use the pre-trained model as it is, extracting different layers representations. We refer to this as the Vanilla BERT model. Some of the popular approaches to extract hidden representations are:

- Using BERT last hidden layer representation
- Concatenation of last 4 BERT layers
- Weighted sum of all 12 layers of BERT
- Using [CLS] tokens as a summary representation for long input sequences.

Investigations ([32]) have shown that the differences between these different approaches are not always explainable. The approach can vary from situation to situation and are often chosen empirically based on performance on a specific task [30; 32]. Some of the feature-based approaches actually were reported to have performance values as good as the fine tuning-based approaches on the Named Entity Recognition task (CoNLL 2003).

In our particular case, the averaging of layers (more than 4) approach was not an efficient solution for us, and was leading to memory overload based on the GPU configuration available to us. This is due to our long 512 token sized sequences, being already demanding on the compute.

[CLS] tokens may not have a semantic meaning on their own, but due to the nature of the pre-training task, [CLS] tokens have been trained to be a representative value for the whole input sequence. It could be hypothesized that a [CLS] token stores information about the average pooling of all the token representations in a sentence (as reported by the authors [1]).

### 2.2.3. Our approach to extracting BERT representations

We combine a BERT fine-tuning approach, with a feature extraction based on [CLS] token of the paper representation.

For our case, based on different experiments, the [CLS] token approach worked best among other feature extraction approaches. We chose to pass gradients through the parameters of the last two layers of the BERT model during our BERT fine-tuning phase. We

[1]Comment by BERT author Devlin https://github.com/google-research/bert/issues/164

chose the last 2 layers to be fine-tuned since empirical experiments have shown that the final layers of BERT are often responsible for the context level meanings, while the initial layers focus more on word meanings. We believe this way, the BERT transformer would fine-tune to learn how to best summarize a sequence, using [CLS] token. In our use case, these text sequences are from research paper abstracts or introduction sections (up to 512 tokens).

We chose to use the BERT base model instead of BERT large which has more parameters, to have quicker runs, and shorter inference time. We load a pre-trained model from hugging face [43] and attempt to fine tune it.[2] The authors reported using 3–5 epochs for most of their fine-tuning tasks [30]. Though larger models would have more model capacity and more expressive representations [30].

We believe out of vocabulary (OOV) terms should not be a problem, as the BERT tokenizer is using a sub-words based tokenizer, which tries to develop context based embeddings for each of the sub words of a word.

**Final layers of BERT have contextual representation**: Ethayarajh [44] state that representations of higher layers store more context specific information.The same results have been observed by many others ([45], [46]), as usually BERT based for downstream tasks only try to fine tune or add to the final layers of BERT.

Interestingly transformer-based models have already been using self-supervised learning objectives for a long time. Though they are not contrastive in nature, like the self-supervised approaches used by us.

### 2.2.4. Using self-supervised frameworks for fine-tuning

Usually, fine-tuning BERT transformers on a downstream task involves using some kind of supervised signal. In our case, we could not afford to fine-tune based on the only available signal, the bids. Bids are not always available for all conferences, and often times they are evoked late in the conference process. Bids themselves could be a noisy assessment of a reviewer's suitability for a submission paper. In addition we wanted to develop a self-supervision approach tailored for expertise modelling. We were looking for ways to obtain embeddings that modelled topic (and less of things like writing style). We turned to self-supervised learning as it fulfilled our above listed criteria and does not need labels.

Self-supervision approaches do not need supervised target labels. Instead, in practice, many self-supervised approaches create pseudo labels from the data itself to start training a model. We chose to work with a Contrastive loss for our approach. We discussed about these approaches in Chapter 1, but as a reminder, a Contrastive loss is a class of losses, that decreases when two samples in a batch are more similar. And this loss increases when the

[2]https://huggingface.co/

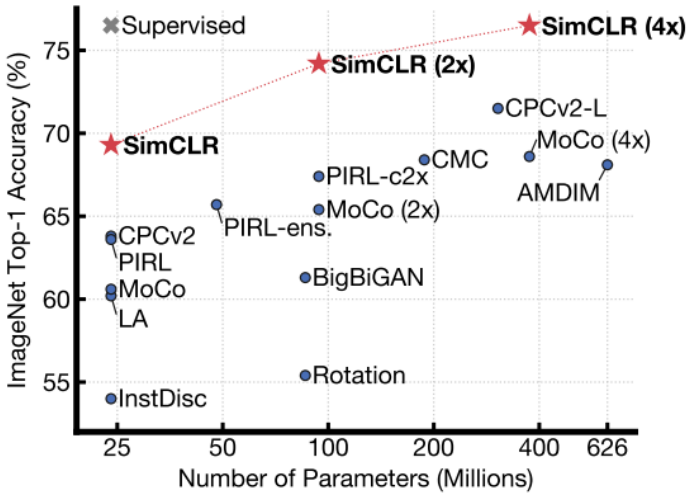



**Figure 2.1.** SimCLR approaches were able to beat the previous state of art result models on computer vision problems (image net) and thus grabbed our attention. The authors Chen et al. [38] asserted that the framework is model agnostic. We have applied self-supervised contrastive learning on textual data in our case. Image is taken from Chen et al. [38]

samples are dis-similar. This is very similar in principle to the triplet-based loss we earlier referred to (see Figure 1.7).

We wanted to fine-tune the BERT transformer in such a way, that it is forced to learn only meaningful things, by adding augmentations to our data and to avoid a model, which overly depends on the style and word-choice of an author to detect a topic, instead of the content itself.

### 2.2.5. Augmentations for self-supervised learning

Augmentations and their choice has been determined to be very important for learning good embedding encoders [38]. Augmentations should try to create transformations that force the model to learn the actual semantic information, instead of superficial changes. For example a human can easily identify that a clockwise rotated image of a dog, and anti-clockwise rotated version of that image, are semantically describing the same thing.

The original SimCLR paper[38] was working with image data. The authors cropped or rotated images, to get different augmented data views. We drew inspiration from these efforts, and realised that expertise should also be identified independent of style. Different reviewers, based on their different backgrounds and origins could have different approaches to describing the same topics. We would like to separate style from content of our textual inputs. These augmentation methods should try to preserve the semantic meaning of the data while removing other style-based elements. To achieve this, we explore different augmentation approaches described below.

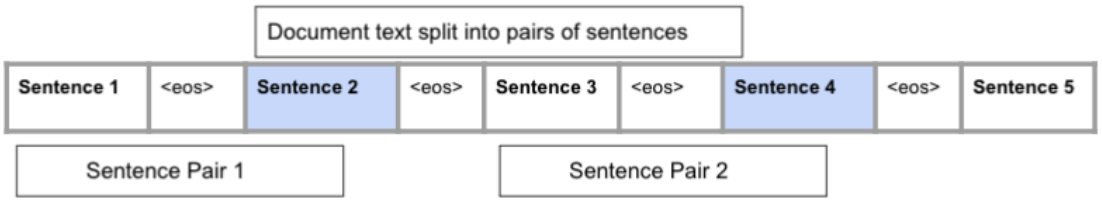



**Figure 2.2.** Next Sentence augmentation task. The BERT representation of the first sentence is matched with the BERT representation of the following sentence.

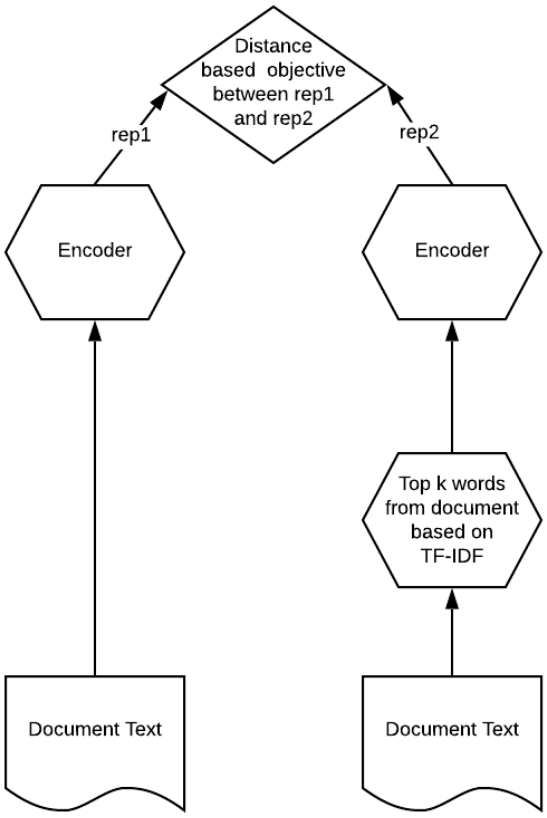



**Figure 2.3.** In all our augmentation approaches for self-supervised learning, the encoder (BERT model) learns to match the representation coming from two inputs, which are basically the same text semantically. Here we demonstrate a Top k BOW augmentation task.

**Retrieving top words from Bag of Words** Figure 2.3 shows a general approach taken by us to implement different augmentation-based approaches. TF-IDF is limited to picking up key words. But we imagined that we could leverage this limitation to pick out words denoting topics. For the Bag of words augmentation task, we first calculated the TF-IDF (Term frequency-inverse document frequency) scores of all the tokens in the corpus of submission papers. TF-IDF scores are generally used to pick up the most salient words


**Figure 2.4.** Bag of words augmentation task. The top portion of this image shows an example text, for which we derived the top 30 tokens as an augmentation. The top 30 words are picked from the text and their BERT representation is made. This BERT representation is then matched with the BERT representation of the original text. We hope to train the model to summarize the text in this way. Notice the words present in the original text at the top and the top words picked up n the transformed text below.

from a given text, based on their frequency in the text corpus. We can imagine that key words or rare words contain the most identifying semantic information for text. A model trained to match the BERT embeddings of the original text, with BERT representation of the top k words from TF-IDF scores, would in effect learn to focus on keywords and prepare a summary representation. We call this approach, the 'BOW augmentation task' for our self-supervised framework. Figure 2.3 shows a top k ($k = 30$) BOW task. We tried with different k values, and $k = 30$ was effective for us, based on observing the loss values during fine-tuning, and accuracy on the validation set. We can postulate that thirty keywords maybe capture enough information about abstracts, where abstracts are usually about 250 to 350 words long.

**Next sentence prediction** This augmentation task is of course inspired by the pre-training tasks already being used for BERT pre-training [30]. We believe that if we train our self-supervised model to match the BERT representation of the first sentence, with the BERT representation of the immediately following sentence, it would force the encoder to learn the salient aspects of research publications. Figure 2.2 shows a *Next sentence augmentation task* example. Usually a next sentence prediction task, implies predicting if a given sentence follows the immediate previous sentence. Due to the nature of scientific papers, all sentences

in an abstract could be having close meanings, and a sentence not following a sentence immediately could still be closely related. It would be a problem for a Contrastive learning-based framework which needs negative samples. Thus at implementation time, to ensure that the self-supervised framework did not overfit, we picked one pair of positive samples as consecutive sentence pairs, and then random sentences picked from other different papers as negative pairs. The Next sentence augmentation task is very useful if we are short on data, since it makes the best use of the available data. For each paragraph, it can be split into multiple pairs of training samples.

**LDA augmentation task**: We have a trained model for LDA representations, using all the text of all submitted papers. This LDA model used unigrams, and hence removes any stylistic info. Any embedding prepared by this LDA model would capture the proportion of conference topics in a given text. We believe that this LDA embedding is a representation in low dimensional topic space and if we trained the model to match BERT's representation of the original text, with its LDA embedding, it would force the model to learn the most important summary information. Also since an LDA model would be aware of all the topics/keywords being talked about across all the other submission papers in the conference, it could be a good way to bring a factor resembling a global parameter factor. BERT representation is squeezed to [dim24] to match the LDA topic vector dimension size of 24. We implemented this augmentation task, by preparing a 24 dimensional LDA vector embedding of the text, and then matching this with a squeezed 24 dimensional (from 768 to 24 dims) BERT representation of the original text.

**Cropping at different noise levels**: In this augmentation task for fine-tuning, we take the text input and then randomly drop 10%, 15%, or 20% of the words in the text. The assumption behind this approach is that dropping a small portion of the words should not change the semantic meaning of the text by a lot. This augmented text is passed through the BERT encoder and then matched with the BERT encoder representation of the original, unaltered text. This approach is inspired by the Masked Language Modelling approach (MLM) Devlin et al. [30]. We hoped that the task of matching the encoder representations would force the model to learn the most important keywords of a text.

**Swapping the order of words**: We swapped the order of words within a sentence and then pass it through the BERT transformer. This augmented representation is to be matched with a BERT representation of the original text. We did controlled experiments with 5%, 10%, or 15% of words swapped. Though we did not obtain success while fine-tuning with this approach. Swapping the word order breaks the contextual semantic of words within a sentence, and is ideally not a good approach for BERT-based models if the goal is to obtain semantic information.

**Matching the representation of Abstract with the Introduction**: We do this augmentation task with the belief that the introduction sections of a research paper, are

at many times, the expanded version of what the research paper would have highlighted in its abstract section. Hence if we try to match the encoder representations between the introduction and abstract, we might learn a 'summarizer' type of encoder, which would be useful for our model.

**2.2.5.1. Using augmentations for Contrastive loss objective.** In the previous section, we explored different augmentation tasks, which could be used to fine-tune a BERT model. Within our self-supervised framework, BERT is referred to as the encoder.

We aimed to train the encoder, along with a projection layer on top (refer to Figure 1.8). The projection layer on top of the BERT transformer encoder squeezes the high dimensional (768 size), to a lower-dimensional vector representation (24 size) upon which we can apply a cosine similarity loss.

We would like to point out to the reader how our approach to contrastive self-supervised learning is similar to earlier approaches based on siamese Networks [47], and triplet loss [36] based distance metric learning approaches. Figure 1.7 illustrates one such structure. Face verification [48] and FaceNet [49], have used similar siamese-based networks in the past to do distance-based learning.

Oord et al. [37] intended to extract useful representations from an encoder trained on general data, and then fine tune using contrastive objective for specific data sets. For example for their NLP task, they used a similar approach to our next sentence prediction augmentation task. They embedded a sentence to a 2400-dimension vector z, followed by a GRU, to predict 3 future sentences using a contrastive loss.

### 2.2.6. From BERT embeddings to score/bid prediction

Till now we have discussed fine-tuning approaches for BERT transformer, using different augmentation tasks for self-supervised frameworks. Once our encoder is ready, we can use it to prepare paper representations for each of the publications individually. The individual paper representations for a reviewer are combined, to predict their bids against a submission paper. The flow of steps is as follows:

- Once the BERT model has trained for sufficient epochs, we can extract the encoder. We mention sufficient steps, as when sufficient data samples are present during the fine-tuning, the training loss showed a continuous downward trend. But one disadvantage of the self-supervised framework is that the performance can be evaluated when we report the final downstream task performance. Keeping this point in mind, we trained at two settings, epochs=50 and epochs=200 for self-supervised fine-tuning of BERT.

- The fine-tuned encoder can then be used to infer papers, both reviewer publications, and submission publications. These BERT embeddings are referred to as the paper representations.
- After the paper representations for all papers of a reviewer, we learn a function to combine the reviewer's corpus of research work. This information is then used along with the submission paper representation, for the bid prediction task.

## 2.3. Reviewer representations

In addition to having paper/document level representations, we also need reviewer-level representations derived from the set of papers they have authored. It is important to be able to combine the different research areas worked on by a researcher. It would be wrong to assume that a researcher would have worked only on a single topic. Some topic areas of researchers might be more represented in their corpus, than others in their publication history. A reviewer could have written several papers during the course of their academic life. Several times, the work that was written originally may not be of current interest to this reviewer.

The important difference we have to consider is that the model for getting reviewer representation, which is a combination of many reviewer papers, and the representation of a submission paper, which is based on a single paper, would be different.

### 2.3.1. Combining text representations of different papers

**2.3.1.1. Naive approach: Averaging reviewer paper representations.** A naive approach to combine different paper representations, would be to do an average of all the individual representations. We believe this is fundamentally problematic as:

- A most basic approach can be an average of all representations. If the different papers of a reviewer are focused on different areas, an average representation is going to lose all uniqueness information and bring them closer to an average value. To experimentally prove our intuitions, we have maintained a Simple Network model in all result tables as a baseline. This simple network is based on a mean of the reviewer paper representations, and is a rough approximation of the current TPMS.
- An averaging approach of reviewer papers is independent of submissions, and stays constant. An average representation could miss out on capturing topics of an individual paper closer to a submission paper, if the other reviewer interest areas are very different. We will introduce in Section 2.3.1.2 a method to conditionally aggregate information from a reviewer's papers based on a submission paper.

We also tried with a max of each dimension and the sum of all dimension approaches, though they significantly under performed against the average approach.

**2.3.1.2. Attention Network.** We opted to model our combination based on an attention network. Initially, we were also tempted by the idea that an attention-based approach could be human interpretable, as the attention scores for each of the reviewer papers could be observed and interpreted.

To condition reviewer papers on submitted papers, we prepare context information. Looking at a fixed submitted paper, $p_i$, attend softly to one of the reviewer papers, $r_j$.

We propose the Attention Network as an attention scoring-based model, which learns to give a score to each of the reviewer's papers, conditioned on its relevance to the bidder submission paper. That is if a reviewer, $r_j$ is to be judged for suitability for a submission paper $p_i$, the model computes $P(paper_{(k,r_j)} \mid p_i)$, where $k$ is some paper in the corpus of reviewer $r_j$.

Figure 2.5 shows the information flow when using such an attention network. To implement attention we opted for the general attention framework as highlighted by Vaswani et al. [28]. We can visualize this as a series of matrix manipulations where the model learns to weight different Key (K) values for a given Query (Q). Each of these values is attained using linear weight matrices, $W_k, W_Q, W_v$.

We experimented with different forms of this equation for our use case. We observed that the value V (obtained from $W_v *$ reviewer paper) in the original general attention did not improve performance. The modified equation which we used for our case:

$$Z = \text{Attention}(Q,K) = softmax(\frac{QK^T}{\sqrt{d_k}}) \tag{2.3.1}$$

We experimentally verified that softmax was an essential component for the model, though changing temperature within the softmax to increase or decrease attention focus did not bring large performance changes. Since attention operates on fixed sized sequences, the dimension of the attention matrix is padded to the largest number of papers written by a reviewer so that the model training in a batch is simplified. Thus a reviewer representation $R_k$ in our formulation, is always weighted with $Z$ of dimension $M$ , where $M$ is the padding value ($Z \in \mathcal{R}^M$):

$$R_k^{\text{weighted}} = Z * R_k \tag{2.3.2}$$

where $R_k^{\text{weighted}} \in \mathcal{R}^P$, and $\mathcal{R}^P$ is the dimension of each of the paper representations $p_i$.

We weigh the different reviewer papers by using the attention scores. $Z$ gives the context weights for each reviewer paper, conditioned on the submission paper which this reviewer is bidding at the moment. Effectively $Z$ is probabilistically choosing papers of the reviewer from all of their corpus, to decide whether they are an expert in the submission paper area

or not. There can be different ways to utilize this $Z$ score, which is a modeling decision. Here are some approaches which we experimented with:

- Concatenate $Z$ with the reviewer papers. In effect, Concatenation would act as a weighting scheme, $R_k^{\text{weighted}} = \text{concat}(Z, R_k)$
- Z scores multiplication with the reviewer papers, ie an inner product $R_k^{\text{weighted}} = Z \cdot R_k$

We observed that the first approach using concatenation worked the best for us in our downstream prediction task.

Equation 2.3.3 describes the components of attention network. For a reviewer paper $r_j$ belonging to a reviewer $R_k$, attention (attn) learns a probabilistic score $z_k$, conditioned against a submission paper $p_i$. Equation 2.3.4 shows how these different components 'add', 'diff' and 'multi' are combined using linear weights and tanh non-linearity. At the end we use a  output with linear weight *combo*, scaled with a scale factor for regression.

We observed through our ablation studies that the add, diff, and multi components were necessary to be able to achieve a good performance. The network is defined as:

$$\begin{aligned} z_k &= \text{attn}(r_j, p_i) \qquad \forall r_j \in R_k, \\ \text{add} &= p_i + (z_k * R_k), \\ \text{diff} &= p_i - (z_k * R_k), \\ \text{multi} &= p_i * (z_k * R_k), \end{aligned} \tag{2.3.3}$$

$$\begin{aligned} \text{combo} &= \tanh(W_{\text{add}} * \text{add}) + \tanh(W_{\text{diff}} * \text{diff}) + \tanh(W_{\text{multi}} * \text{multi}), \\ \text{out} &= \text{scaling} * W_{combo}(\sigma(\text{combo})) \end{aligned} \tag{2.3.4}$$

Through ablation studies, we observed that the tanh nonlinearity is essential for modeling. We keep the scale factor at the end as 3, since the bid values are between (0,3) for both the data sets.

Our experiments show that this model gave the best performance amongst all possible models when used with LDA embeddings. We postulate that the low 25 dimensional embeddings of LDA captured a lot of useful semantic information, and it was much more suited to the relatively small-sized data available to us.

**2.3.1.3. Simple linear network.** A simple linear network uses the mean of reviewer paper representations. Equation 2.3.5 shows our formulation, where a reviewer $R_k$, could have $M$ papers (or padded till $M$ number of papers) $r_j$ ($R_k \in \mathcal{R}^M$). This is our baseline comparison model for reviewer expertise. It can also be seen as a rough approximation of the current TPMS model for reviewer representation.

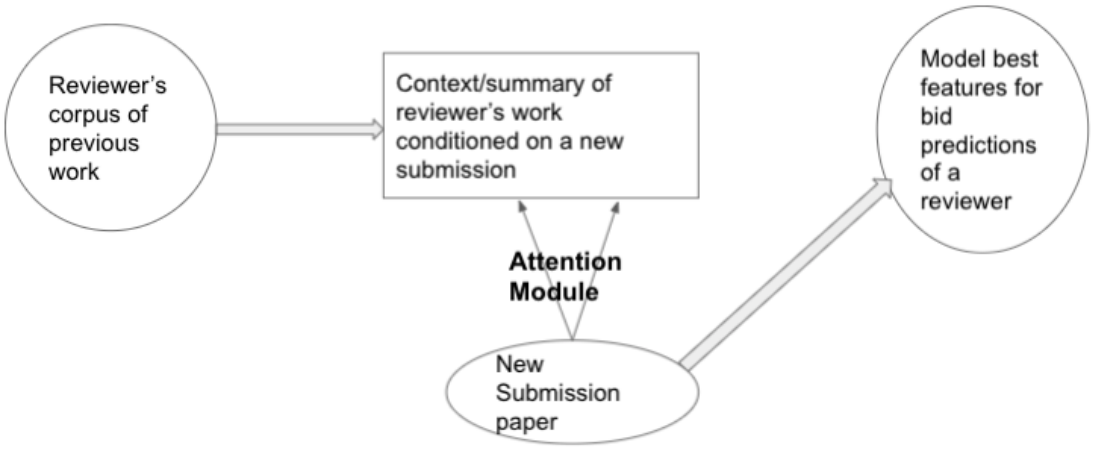



**Figure 2.5.** Flow of information for the Attention Network. Each of the reviewer papers gets an attention score, conditioned on the submission paper they are bidding on.

We postulate that since averaging would nullify all the uniqueness of representations, this model should perform poorer than an attention-based model.

Simple Linear Network uses the 768-dimensional BERT encoder embeddings. This network is composed of tanh non-linearity composed with Linear weights and a sigmoid at the end. We experimentally observed that tanh non-linearity was essential for the model to work, and hence this is the simplest possible baseline model using BERT embeddings.

$$\begin{aligned} \text{add} &= p_i + \frac{1}{M}\sum_j (r_j), \qquad\qquad r_j \in R_k, \\ \text{diff} &= p_i - \frac{1}{M}\sum_j (r_j), \\ \text{combo} &= \tanh(\text{add}) + \tanh(\text{diff}), \\ \text{out} &= \text{scaling} * W_{combo}(\sigma(combo)) \end{aligned} \tag{2.3.5}$$

### 2.3.2. Loss objective for the bid prediction task

We model ***the bid prediction problem*** as a regression. In this work of ours, we often refer to this bid prediction task as ***the downstream task*** since we have kept the paper representation learning and prediction problems separate. The bid prediction task tries to anticipate the bid, which a reviewer would give on a submission paper, based on their self-attested expertise and interest.

The neural network model is going to predict a continuous value within the range of bid values, using a scale factor. As is standard for regression, we use a mean squared error loss

(Equation 2.3.6) as the objective:

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.3.6}$$

where $y_i$ is the actual bid value, and $\hat{y}_i$ its prediction.

## 2.4. Datasets

We introduce two datasets used in our experiments, NeurIPS 2019 and PLDI 2021.

### 2.4.1. NeurIPS 2019

NeurIPS (Neural Information Processing Systems) is a scientific conference on artificial intelligence and machine learning, held each year during December. It is considered one of the most prestigious in its field, and is at the forefront of AI/ML research. We had access to the reviewer corpus, submission papers and reviewer bids for the NeurIPS 2019 conference.

For the NeurIPS 2019 conference, each of the reviewers can bid on a submission paper based on their interests and expertise. The bid values can be 0,1,2,3, with the following interpretations:

- 0: Not willing to review.
- 1: Not willing, unless absolutely unavoidable.
- 2: Willing to review.
- 3: Eager to review

Though we treat these bid values as self-attested expertise elicitation, than interest alone.

When we train a model for the downstream task each sample is a reviewer, submission paper pair, where a bid has been recorded.

Figure 2.6 shows the distribution of bids. We noticed that bids are skewed towards 0. On further investigation, we found that there were a large number of reviewers giving 0 bid values on more than 300 papers at a time. This can be seen as misunderstandings at the end of reviewers, about the meaning of bids.

Upon removing the bids valued at $-2$ (1 bid) and $-1$ (1521 bids) as they are not valid values for TPMS, we get the following distribution.

(1) Number of reviewers: 3961
(2) Number of submission papers: 6810
(3) Number of the reviewer, submission bid pairs 788371

Interestingly we observed that 0 is the most common bid value used by reviewers. We observed that 98 reviewers bid 0 on more than 5,000 submission papers. Other than this set of reviewers, we hypothesise that maybe most reviewers were very particular about not getting assigned papers they were not interested in, and 0 bid was a way to ensure that.

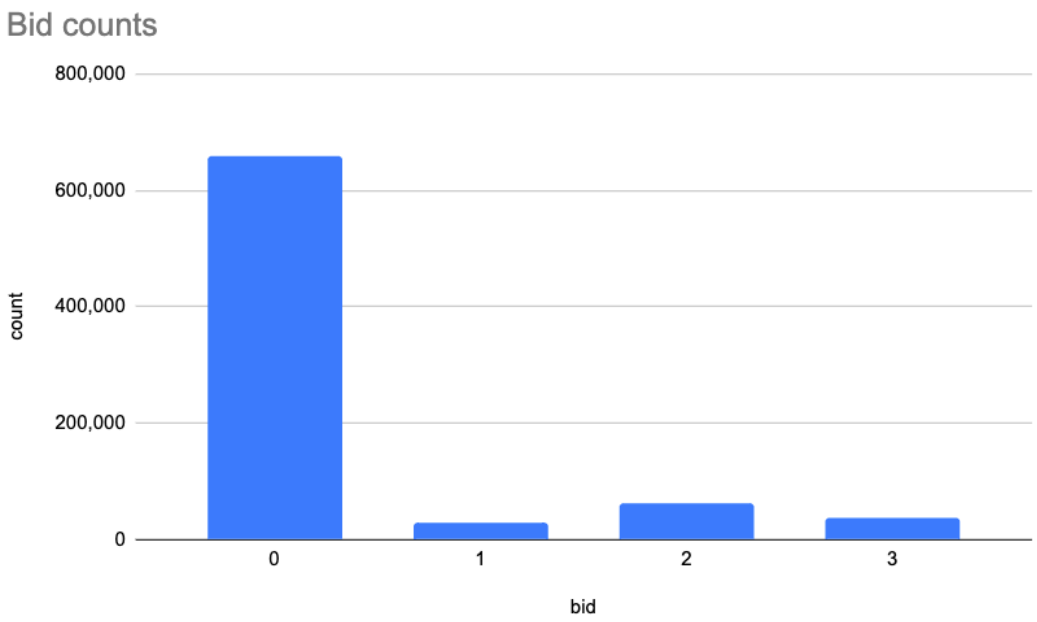



**Figure 2.6.** Bids distribution originally for NeurIPS 2019. Reviewers could express their interest in a submission paper by bidding 0,1,2, or 3 There is a heavy skew towards 0 bids in comparison to other bid values.

**2.4.1.1. Data pre-processing decisions: Use of area chair data.** When we first trained our downstream model for the bid prediction task, on the original NeurIPS 2019 dataset, we could not achieve any success. Attempts to model bid prediction failed on this original dataset. Previous work [6] had highlighted the same by pointing out that bids are inherently noisy as (1) it is difficult for reviewers to offer reasonable assessments of all but a small fraction of the papers, given the large numbers involved and (2) reviewers usually have access to only title and abstract, which is very less information to make a judgment. We also found that not all reviewers have the same understanding of what bid values 0, 1,2, or 3 mean. Hence leading to even more noise.

We then decided to use only area chairs (AC) bids data. Just using AC data improved performance drastically. For 332 area chair reviewers (AC), we had 67k data points (bids on papers by reviewers). The distribution of bids, in this case, is shown in Table 2.1.

| Bid value | Percent share of Bids data |
|---|---|
| 0 | 75% |
| 1 | 3% |
| 2 | 14% |
| 3 | 6 % |

**Table 2.1.** Bid counts after filtering only for area chair bids, for NeurIPS 2019 dataset

We postulate that area chairs being senior members behave more responsibly in the bidding process. If they notice something in their area of expertise, they would bid highly. If they are aware that a submission paper is out of their area of expertise they do not bid on it. Hence this makes the data less noisy, and easier to learn, if a model is trying to predict bids based on the assumption that people bid only on areas that they know about.

The final description for NeurIPS 2019 dataset is :

(1) Number of reviewers: 300

(2) Number of submission papers: 6810

(3) Number of the reviewer, submission bid pairs 49380

For our BERT transformer fine-tuning, we were able to use the 6810 submission papers. While at the downstream bid prediction task, we had 49,000 data points.

### 2.4.2. PLDI 2021 dataset

PLDI is a conference for programming languages and programming systems research.[3] The dataset we have available is for PLDI 2021. For this conference we have 70 reviewers and 1000 submission papers. Though we had access to reviewers corpus for only 67 reviewers.

We explored the dataset and saw that the distribution of bids is between -100 to 100. For this conference, -100 implies the least possible interest in a submission paper by a reviewer, and 100 reflects the highest possible interest. As can be seen from the Figure 2.7, data is skewed heavily towards the negative side, implying people bid proactively to avoid getting assigned papers they are not interested in.

After some pre-processing decisions, the final dataset which we used for training the model on our downstream task had the following statistics:

(1) Number of reviewers 67

(2) Number of submission papers 320

(3) Number of reviewer, submission bid pairs 5500

We can notice that this dataset is very small in comparison to the NeurIPS 2019 dataset, where we had about 10 times more data. This in turn affected our ability to fine-tune the BERT transformer.

**2.4.2.1. Bid normalization.** In order to use the same two downstream network architectures as the NeurIPS 2019 dataset, we normalized the bids data within the $0 - 3$ bid values range in two steps.

In the first step, we wanted to make the data more symmetric. Upon data observation, it can be seen from Figure 2.7, that the $-20$ bid value is the most represented negative bid value, implying most reviewers (except a very few) did not have the heart to rate $-100$ on a submission paper. We hypothesize that through an informal agreement, reviewers may be

[3] https://conf.researchr.org/home/pldi-2021

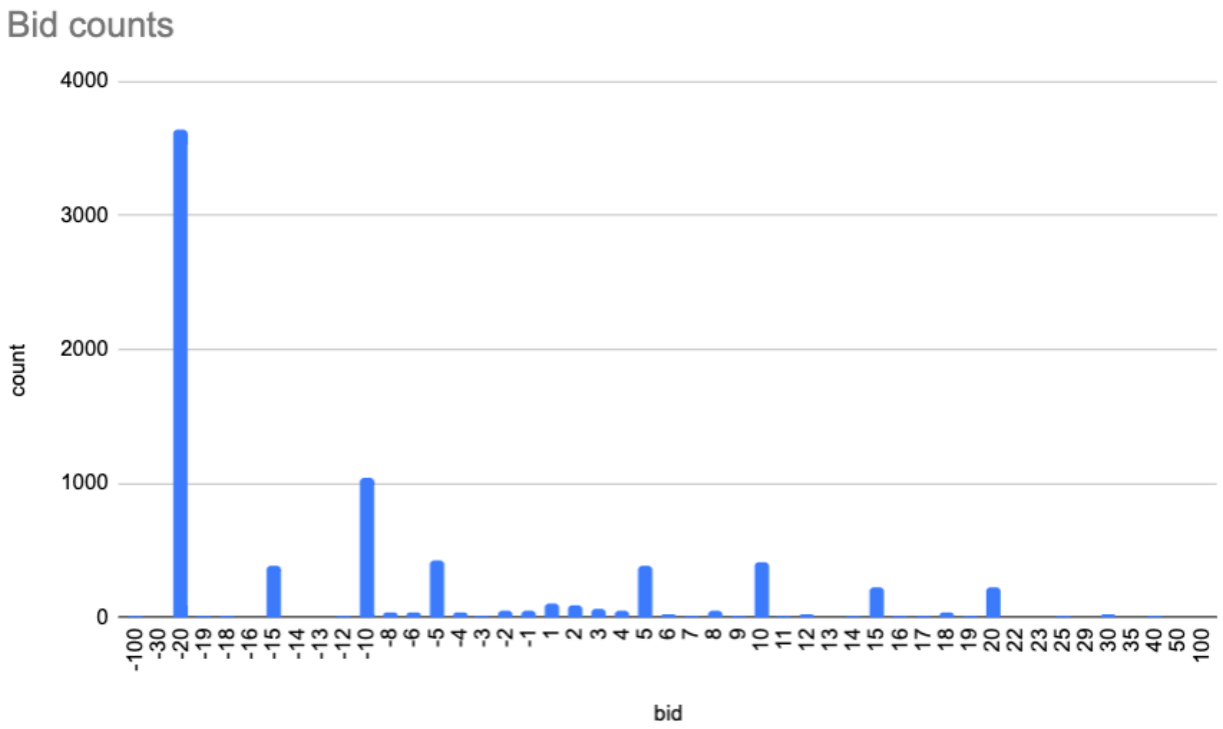



**Figure 2.7.** Bids distribution PLDI 2021. Reviewers were free to bid from −100 to 100, to express their interest on a submission paper. −20 bid is the most represented negative bid value, whereas on the positive side there is no clear peak.

agreed to bid −20 to show their disinterest. We then transformed all values less than −20 to −20. On the positive side we notice that beyond 20 bid values, other bid values are sparse. Hence we restricted the 20 bid value, as the bid value reflecting the most interest.

$$\text{Bid}_{Normalized} = 3 * \frac{(\text{Bid} - \text{Bid}_{min})}{(\text{Bid}_{max} - \text{Bid}_{min})} \tag{2.4.1}$$

## 2.5. Experimental Results

To judge the performance of our self-supervised models, we can only measure their performance on the bid prediction task. This is consistent with the self-supervised literature which tests the performance of models on their chosen downstream tasks [38]. Hence all results are reported together as a combination of encoder model (BERT fine-tuned encoder using an augmentation task) combined with downstream prediction network.

**Baseline scores**: The naive baseline score which we have is, the MSE obtained when a bid prediction model always predicts the average bid value or the majority class. We also report results from a Vanilla BERT model. This is a pre-trained BERT transformer, with frozen parameters, used as it is from the Hugging Face library [43].[4]

[4]https://huggingface.co/

### 2.5.1. Results on the downstream bid prediction task

In this section we discuss our results on the downstream bid prediction task for different setups. Each experimental setup is a combination of:

- Choice of encoder: How the chosen encoder was fine-tuned on the self supervised framework. The chosen encoder is used for generating paper representations.
- Choice of the bid prediction model: We can use either a Simple Network, or the Attention Network

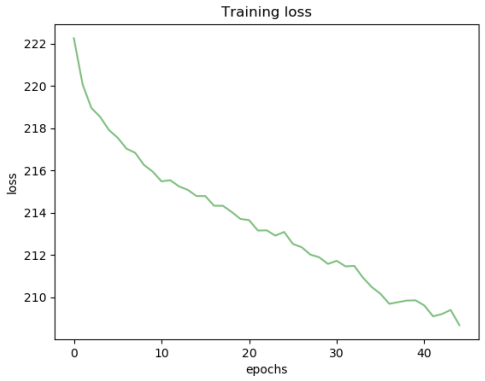
Since we can not report results on the self-supervised fine-tuning step of BERT, we report MSE loss values on the downstream bid prediction task as the final metric. To have a competitive baseline, apart from Vanilla BERT, we also include the SciBERT (Beltagy et al. [31]) in our result tables. All these results are summarized in tables, Table 2.2, Table 2.3, Table 2.7 and Table 2.6. In general we observed across these results, that a combination of any encoder with Attention Network always beats the combination of any encoder with a Simple Network.

**2.5.1.1. Results on NeurIPS 2019.** NeurIPS 2019 data set has around 7,000 submission papers available. These were used as our corpus for fine-tuning the BERT transformer model. In this section we first discuss training the Bert encoder using different augmentation approaches, and then move on to demonstrate their performance on the bid prediction task.

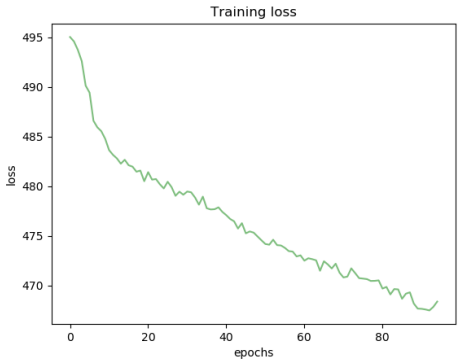
We noticed that we were able to fine-tune the next sentence augmentation task BERT model till 200 epochs, with a continuous decrease in the training loss. The next sentence augmentation task is able to perform the best (Tables 2.3 , 2.2), since it makes the best use of the available data, converting 7000 submission papers, to 30,000 training samples.

Among all the augmentation task approaches for BERT fine-tuning, we observe that the Attention Network consistently outperforms the Simple Network. This reinforces our belief that combining the reviewer representations from different papers is important, and can not be done by just using an average representation, like for a simple network.

For the NeurIPS 2019 data set, LDA augmentation task was able to beat the Vanilla BERT and Simple network, even in the BYOL self-supervised framework (refer to Table 2.4). But this is not the case for the PLDI 2021 dataset. We believe that this could be related to the size of data available for fine-tuning. For the LDA augmentation task, we have 6,810 training samples available at the fine-tuning time (NeurIPS 2019 dataset). We believed that since an LDA model had access to all submission papers during training, trying to match an LDA representation would bring information about the whole conference. Though the representation model was not able to beat the Vanilla BERT model with an Attention Network, it did beat the Vanilla BERT representations under the simple network model. To further confirm the results, we also tried a BYOL [42] based encoder.

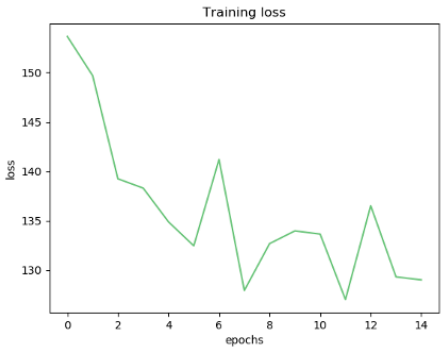



(a) Train loss curve for next sentence augmentation task

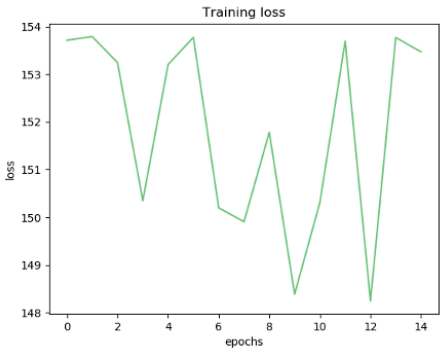



(b) Train loss curve for LDA augmentation task

**Figure 2.8.** Training curves for these two augmentation approaches worked well. The loss curve went down smoothly for up to 200 epochs, which we then tested on downstream task. We did not achieve much success with swap or crop augmentation tasks .Encoder fine tuned using SimCLR framework on NeurIPS 2019 dataset.




(a) Train loss BOW




(b) Train loss intro abs

**Figure 2.9.** Training curves which did not train well for fine-tuning BERT on the PLDI 2021 dataset. We posit that this is due to the very small size of submission corpus for this conference. This is also reflected in the performance of BOW augmentation and intro-abs task encoder on downstream prediction task at Table 2.7.

The swap order augmentation task was not very fruitful for fine-tuning. We can hypothesis that BERT transformers derive their power from their efficient use of word order and contextual word embeddings. Hence a swap order which effectively destroys the original sentence semantics can not be a useful model to train an encoder for our case.

Table 2.2 shows results on downstream tasks using a Simple Network. We can observe that the LDA augmentation task and Next sentence augmentation task were able to beat the Vanilla BERT embeddings in this scenario. We also observed that it was difficult to search the

correct hyper-parameters for results with SciBERT combined with Simple Network (Table 2.2).

| Encoder model with **Simple Network** | Validation loss | Test loss |
|---|---|---|
| **Naive bidding** | 0.94 | 0.94 |
| **Vanilla BERT** | 0.51 | 0.53 |
| **SciBERT** | 0.93 | 0.95 |
| **LDA augmentation BERT** | 0.42 | 0.44 |
| **Next Sentence augmentation BERT** | 0.48 | 0.49 |
| **Bag of Words augmentation BERT** | 0.61 | 0.61 |
| **Intro and abstract match augmentation BERT** | 0.60 | 0.60 |

**Table 2.2.** Results on NeurIPS 2019 dataset with Simple Network. Encoder fine-tuned using SimCLR framework. We can observe that the LDA augmentation task and Next sentence augmentation task were able to beat the Vanilla BERT embeddings.

Table 2.3 shows results on downstream tasks using Attention Network. 200 epochs in the name LDA augmentation refers to 200 epochs of self-supervised fine-tuning on the BERT transformer. We were able to beat the Vanilla BERT model in this case by a small margin.

The table 2.4 shows BYOL fine-tuned BERT encoder used with the downstream task. The BYOL framework for self-supervised learning could be an interesting future direction of work since it does not depend on having *negative samples* in each batch in contrast to other self-supervised learning approaches.

**2.5.1.2. Results on PLDI 2021 dataset.** PLDI 2021 is a much smaller conference compared to NeurIPS 2019. Due to having only 320 submission papers in the dataset, fine-tuning a transformer model on it is not a very effective task. The results in tables, Table 2.7 and Table 2.6, show that attention network consistently outperforms a simple network. Though apart from the next sentence augmentation task, we could not achieve results better than the Vanilla BERT encoder.

Table 2.7 shows results on downstream tasks using Simple Network.

Table 2.6 shows results on downstream tasks using Attention Network. If we compare these results with Table 2.7, we can see that Attention Network has consistently achieved much lower MSE values. Though Vanilla BERT has remained the best representation among

| Encoder model with **Attention Net** | Validation loss | Test loss |
|---|---|---|
| **Naive bidding** | 0.94 | 0.94 |
| **Vanilla BERT** | 0.19 | **0.19** ±0.005 |
| **SciBERT** | 0.17 | **0.175** ±0.010 |
| **LDA augmentation BERT** | 0.23 | 0.23 ±0.015 |
| **LDA augmentation 200 epochs BERT** | 0.20 | 0.21 |
| **Next Sentence augmentation BERT** | 0.20 | 0.21 ±0.02 |
| **Next Sentence augmentation 200 epochs BERT** | 0.18 | **0.18** ±0.002 |
| **Bag of Words augmentation BERT** | 0.21 | 0.21 ±0 |
| **Intro and abstract match augmentation BERT** | 0.21 | 0.21 |

**Table 2.3.** Results on NeurIPS 2019 dataset with Attention Network. Encoder fine-tuned using SimCLR framework. The best reported values are in bold. Notice that all values are much better than those achieved by the Simple Net. Note: 200 epochs in the name LDA augmentation refers to 200 epochs of fine-tuning on the BERT transformer, done before training on the bid prediction task.

| Encoder model | Downstream model | Validation loss | Test loss |
|---|---|---|---|
| - | **Naive bidding** | 0.94 | 0.94 |
| **Vanilla BERT** | **Simple Net** | 0.73 | 0.68 |
| **LDA augmentation BERT** | **Simple Net** | 0.61 | 0.63 |

**Table 2.4.** Results on NeurIPS 2019 dataset with encoder fine-tuned using BYOL framework. LDA augmentation task was able to beat the Vanilla BERT embeddings.

all the encoders, suggesting that fine-tuning transformers on a very small dataset are not sufficient

Based on our observation of performance differences between NeurIPS 2019 (Section 2.5.1.1) and PLDI 2021 (Section 2.5.1.2), we can see that the next sentence augmentation approach is a safe choice for paper representation. It makes the best use of available data, and

is suitable for smaller conferences (or other text corpus). In the presence of larger datasets, other approaches such as an LDA augmentation task could be more useful. This could be an important decision factor when choosing a fine-tuning approach for future conferences.

### 2.5.2. Decision to use Abstract texts

In the previous two sections we talked about results on bid prediction task using different combinations of representations and downstream models. At this stage we would like to point out to the reader, that previous results are reported using representations built from abstract sections of the papers. The question which arises then would be, what would the performance if we decided to use text from introduction sections instead.

In this section we explore how much of a difference does it make if we have access to text either only from abstract, or introduction, or both introduction and abstract. The results in Table 2.5 indicate that abstracts are usually well-written summaries of whole papers, and hence we are not missing out on a lot of information by just using them. Also important to remember is that the BERT model is itself limited to 512 tokens in a sequence.

Ideally, we would want to be able to use the full text of a submission paper for preparing an accurate representation of a paper. But we can not do that due to two reasons:

- Many conferences start scoring reviewer-submission pairs when the full paper has not been submitted yet.
- BERT model used for our implementation is limited to 512 tokens in total. Data observation showed us that most abstracts were between 250 - 300 words.

Table 2.5 compares the downstream task performance when using

- Only abstract text of the papers
- Using only introduction section text (512 words max)
- Using text from both introduction and abstract. Though using the full text from both sections is practically not possible due to the BERT tokens limit.

While using the third approach, effectively we used only $\approx$ 200 initial words from an introduction since BERT models have a hard limit for 512 tokens, and we would have already parsed $\approx$ 250 input tokens from the abstract. We can see that using only the abstract did not lead to loss of performance on the downstream task, and in fact, the three approaches have very similar results. We postulate that since an abstract is prepared by researchers to be a very apt summary of their paper, hence most of the useful information has been already condensed there.

Table 2.6 is showing results on downstream tasks using Attention Network. Table 2.7 shows the results on downstream tasks for the Simple Network. We can notice clearly that the Attention Network is better over the Simple Network in this case over all the possible BERT fine-tuning approaches.

| Encoder model with Simple Net | Encoded text | Test loss |
|---|---|---|
| - | **Naive bidding** | 0.94 |
| **Vanilla BERT** | only abstract | 0.69 |
| **Vanilla BERT** | only introduction | 0.72 |
| **Vanilla BERT** | both introduction and abstract | 0.70 |

**Table 2.5.** Comparing the use of abstract, introduction, or both intro and abstract for learning text representation. Results on NeurIPS 2019 dataset with encoder fine-tuned using SimCLR framework.

| Encoder model with **Attention Net** | Validation loss | Test loss |
|---|---|---|
| **Naive bidding** | 0.87 | 0.87 |
| **Vanilla BERT** | 0.45 | 0.46 $\pm$0.04 |
| **SciBERT** | 0.46 | 0.46 |
| **Next Sentence augmentation BERT** | 0.60 | 0.59 $\pm$0.045 |
| **Bag of Words augmentation BERT** | 0.68 | 0.64 $\pm$0.05 |
| **Intro and abstract match augmentation BERT** | 0.72 | 0.67 $\pm$0.07 |

**Table 2.6.** Results on PLDI 2021 dataset with Attention Network. Encoder fine-tuned using SimCLR framework. If we compare these results with Table 2.7, we can see that Attention Network has achieved much better values. Though Vanilla BERT remains the best representation among all, suggesting that fine tuning transformers on a very small dataset is not sufficient.

### 2.5.3. Ablation studies on the Attention Network

**2.5.3.1. What makes Attention Network work?** For the ablation study, we chose to focus on low dimensional representations/embeddings namely LDA, which due to their simpler nature are easier to interpret. The values of Naive bidding model and BOW representation are reported as baseline reference numbers.

Table 2.8 shows the different combinations we tried. These are of the form embedding method/paper representation and downstream prediction model.

| Encoder model with **Simple Net** | Validation loss | Test loss |
|---|---|---|
| **Naive bidding** | 0.87 | 0.87 |
| **Vanilla BERT** | 0.73 | 0.68 ±0.045 |
| **SciBERT** | 0.93 | 0.94 |
| **Next Sentence augmentation BERT** | 0.73 | 0.70 ±0.02 |
| **Bag of Words augmentation BERT** | 0.67 | 0.67 ±0.03 |
| **Intro and abstract match augmentation BERT** | 0.68 | 0.67 ±0.045 |

**Table 2.7.** Results on PLDI 2021 dataset with Simple Network. Encoder fine-tuned using SimCLR framework.

We did ablation testing with the different components of the attention network namely : add, diff and multi (we discussed these components in Section 2.3.1.2). We had postulated that these components capture different scenarios of reviewers. Table 2.8 shows that the 'add' and 'diff' components are most essential for the attention network, along with non-linear activation function. We did our ablation studies while keeping a check on the MSE test loss, the test accuracy of prediction, and the confusion matrix of prediction on different bid values.

For a BOW paper representation approach, we learned the term frequency-inverse document frequency values (TF-IDF), using the corpus of submitted papers. Then we prepared BOW representations for all reviewer and submission papers involved.

We had an intuition that LDA models were going to be better, as an LDA latent topic space is a better summary representation than a BOW embedding, which uses just a keyword frequency counting. For BOW representations, the representation vectors are dimensionally very large (dim 22000 for NeurIPS 2019), as the size of representation depends on the size of vocabulary learned from the corpus.

We hypothesise that the attention network may be most advantageous when there is an expert who has diverse interests. Attention could enable the model to focus on relevant areas/papers instead of looking at all the papers of a reviewer.

As a baseline measure, we first ran the model, using BOW representations for the documents, combined with a Simple linear Network (referred here as Simple Net). We saw that the MSE test loss was the worst possible, i.e. close to a naive prediction model. Upon investigating the confusion matrix of bid prediction values, we saw that the model just learned to predict 0 all the time, since 0 was the most represented data point.

| Embedding approach with downstream model | MSE(test loss) | comments |
|---|---|---|
| **Naive bidding** | 0.940 | -. |
| **BOW Simple Net** | 0.930 | Model overfits and predicts majority class bid 0 everywhere. |
| Using LDA embeddings with different downstream models | - | |
| **Simple Net** (uses dot product reviewer paper and submission paper) | 0.837 | Model does not predict majority class every time. Better MSE than BOW. |
| **Simple Net** (concatenate reviewer representation with submission representation) | 0.640 | - |
| **Attention Net** (without multi term) | 0.195 | Best combination. |
| **Attention Net** (without Add, Subtract terms. only multi term) | 0.337 | - |
| **Attention Net** (without activation function) | 0.430 | - |
| **Attention Net** (no softmax in attention module) | 0.292 | Softmax essential to keep all attention scores in the meaningful $0-1$ range. |
| **Attention Net** | 0.213 | All components used of Attention Net used. |

**Table 2.8.** Ablation over components present in the Attention Network, NeurIPS 2019 dataset. This table presents results using the simpler LDA and BOW embeddings without any use of BERT or BERT embeddings. Note that this ablation study is based on simple LDA and BOW representations, instead of any BERT based fine-tuned encoders.

As we had expected, the LDA representations and attention network combination was easily able to beat the BOW representations and simple network baseline.

At the onset, we were already aware that LDA representations are a better approach for paper representation than BOW. Results from Table 2.8, show that the best performance on MSE test loss is achieved when using LDA embeddings with our proposed attention network without the multiplication term. LDA embeddings with the full Attention Network also gave a reasonably low MSE test loss value. In contrast, when we used just the multiplication term in the attention network, we still observed a low enough MSE test loss, which hints that multi-term brings non-linearity essential for learning reviewer and submission interest scores.

**2.5.3.2. Where does Attention Network focus?** Figure 2.10, shows the attention scores spread out over all papers of a reviewer. The reviewer in Figure 2.10 had only 15

papers, reflected by low attention scores after $15^{th}$ index. Since attention operates on fixed length sequences, we had fixed the size of the attention matrix at 122 (this was the maximum number of papers written by a reviewer in the dataset). Upon qualitatively inspecting the index showing the highest attention score, we observed that the model was attending to this reviewer paper based on word overlaps with the submission paper. Ideally, this could be a clean interpretation of how attention is working, but it is not always the case. Sometimes attention scores were high for a reviewer paper, based on no apparent similarity, or equally split across all reviewer papers. This could be an interesting direction of work for future to see which are the problem cases, but we did not investigate this direction any further.

```
tensor([2.2463, 2.2298, 2.1787, 1.4259, 2.8604, 0.6626, 0.7672, 0.7491, 0.6979,
        2.4269, 0.7327, 0.6589, 1.6233, 2.2300, 0.6845, 0.8737, 0.8737, 0.8737,
        0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737,
        0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737,
        0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737, 0.8737,
```

**Figure 2.10.** Attention scores over a reviewer's set of papers. The scores are conditioned based on the submission paper and hence imply an individual matching score. This reviewer has high scores for the first five and tenth papers, while the rest of the papers have low relevance for the new submission.

# Chapter 3

# Conclusion and future directions

## 3.1. Conclusion

We showed during the course of our work on choice of paper representations and encoder that a better paper representation can improve the suitability prediction of reviewers on a submission paper. We were able to fine-tune a BERT transformer model to our corpus of conference submission papers, with a self-supervised framework.

This self-supervised framework removes our dependence on bids as signals for fine-tuning the BERT encoder. We also showed that the combination of different paper representations is an important problem area to solve, and proposed an attention network conditioned on the new submission papers, to build an expertise model.

Our results showed that the attention network outperforms the simple network-based model. We explored several approaches for augmentations in the self-supervised framework and were able to outperform the Vanilla BERT with attention network, by our next sentence augmentation BERT with attention network combination (for NeurIPS 2019 dataset). Though the rest of our augmentation approaches failed to achieve similar results. The next sentence augmentation could be suitable for making the best use of available data for fine-tuning.

## 3.2. Future Work

One key challenge we felt while working with the NeurIPS 2019 dataset was noise in the bids data. This could have happened due to different interpretations of reviewers about the scale of bid values. We had to resort to using the area chair bids, to tackle this problem, but in the future, there could be a model which is able to calibrate all these differently interpreted bids to a single scale.

In this work, we stuck to using BERT as the transformer of choice. Though we are aware that BERT has its limitations when it comes to modeling longer text sequences. It would

be interesting to see the performance of transformers more suitable for longer text sequences such as Reformers [50] or Big Bird [51].

Another approach which could be explored in future would be to experiment with recent tokenization approaches, which work with vocabulary of multiple languages at the same time such as SentencePiece ([52]). A tokenizer capable of efficiently learning the most important words in a corpus, would have benefits in paper representation.

Some other areas which can be explored would be making a more explainable system. Currently how the attention network chooses its scores is not very explainable. Finding out the limitations and advantages of this combination would be a good way to control for issues about fairness arising in the future.

# Références bibliographiques


[1] Laurent Charlin and Richard Zemel. The toronto paper matching system: an automated paper-reviewer assignment system. 2013.

[2] Ziming Zhuang, Ergin Elmacioglu, Dongwon Lee, and C Lee Giles. Measuring conference quality by mining program committee characteristics. In *Proceedings of the 7th ACM/IEEE-CS joint conference on Digital libraries*, pages 225–234, 2007.

[3] Andrew Tomkins, Min Zhang, and William D Heavlin. Reviewer bias in single-versus double-blind peer review. *Proceedings of the National Academy of Sciences*, 114(48): 12708–12713, 2017.

[4] Robert K Merton. The matthew effect in science: The reward and communication systems of science are considered. *Science*, 159(3810):56–63, 1968.

[5] Mark Allman. What ought a program committee to do? *WOWCS*, 8:1–5, 2008.

[6] Laurent Charlin, Richard S Zemel, and Craig Boutilier. A framework for optimizing paper matching. *arXiv preprint arXiv:1202.3706*, 2012.

[7] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.

[8] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

[9] Ivan Stelmakh, Nihar B Shah, and Aarti Singh. Peerreview4all: Fair and accurate reviewer assignment in peer review. In *Algorithmic Learning Theory*, pages 828–856. PMLR, 2019.

[10] Philippe Rigaux. An iterative rating method: Application to web-based conference management. In *Proceedings of the 2004 ACM symposium on Applied computing*, pages 1682–1687, 2004.

[11] Don Conry, Yehuda Koren, and Naren Ramakrishnan. Recommender systems for the conference paper assignment problem. In *Proceedings of the third ACM conference on Recommender systems*, pages 357–360, 2009.

[12] Marko A Rodriguez and Johan Bollen. An algorithm to determine peer-reviewers. In *Proceedings of the 17th ACM conference on Information and knowledge management*,

pages 319–328, 2008.

[13] David Mimno and Andrew McCallum. Expertise modeling for matching papers with reviewers. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 500–509, 2007.

[14] Krisztian Balog, Leif Azzopardi, and Maarten De Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50, 2006.

[15] Dawit Yimam-Seid and Alfred Kobsa. Expert-finding systems for organizations: Problem and domain analysis and the demoir approach. *Journal of Organizational Computing and Electronic Commerce*, 13(1):1–24, 2003.

[16] Henry Kautz, Bart Selman, Al Milewski, et al. Agent amplified communication. In *AAAI/IAAI, Vol. 1*, pages 3–9, 1996.

[17] Fangqiu Han, Shulong Tan, Huan Sun, Mudhakar Srivatsa, Deng Cai, and Xifeng Yan. Distributed representations of expertise. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 531–539. SIAM, 2016.

[18] Yujie Qian, Jie Tang, and Kan Wu. Weakly learning to match experts in online community. *arXiv preprint arXiv:1611.04363*, 2016.

[19] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009.

[20] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR, 2021.

[21] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[22] Bruce Croft and John Lafferty. *Language modeling for information retrieval*, volume 13. Springer Science & Business Media, 2003.

[23] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[24] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196. PMLR, 2014.

[25] Chia-Wei Liu, Ryan Lowe, Iulian V Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. *arXiv preprint arXiv:1603.08023*, 2016.

[26] Andrew M Dai, Christopher Olah, and Quoc V Le. Document embedding with paragraph vectors. *arXiv preprint arXiv:1507.07998*, 2015.

[27] Minmin Chen. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377*, 2017.

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

[29] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

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

[31] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676*, 2019.

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

[33] Guillaume Lample and Alexis Conneau. Cross-lingual language model pretraining. *arXiv preprint arXiv:1901.07291*, 2019.

[34] Gabriel Forgues, Joelle Pineau, Jean-Marie Larchevêque, and Réal Tremblay. Bootstrapping dialog systems with word embeddings. In *Nips, modern machine learning and natural language processing workshop*, volume 2, 2014.

[35] Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*, 2019.

[36] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International workshop on similarity-based pattern recognition*, pages 84–92. Springer, 2015.

[37] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[38] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[39] Paras Jain, Ajay Jain, Tianjun Zhang, Pieter Abbeel, Joseph E Gonzalez, and Ion Stoica. Contrastive code representation learning. *arXiv preprint arXiv:2007.04973*, 2020.

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[41] Philip Bachman, R Devon Hjelm, and William Buchwalter. Learning representations by maximizing mutual information across views. *arXiv preprint arXiv:1906.00910*, 2019.

[42] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[43] 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. 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, October 2020. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/2020.emnlp-demos.6`.

[44] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[45] Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A Gers. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1823–1832, 2019.

[46] Anna Rogers, Olga Kovaleva, and Anna Rumshisky. A primer in bertology: What we know about how bert works. *Transactions of the Association for Computational Linguistics*, 8:842–866, 2020.

[47] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[48] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546. IEEE, 2005.

[49] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[50] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[51] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. In *NeurIPS*, 2020.

[52] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.

| **Ordre des éléments constitutifs du mémoire ou de la thèse** | | |
|---|---|---|
| 1. | La page de titre | obligatoire |
| 2. | La page d'identification des membres du jury | obligatoire |
| 3. | Le résumé en français et les mots clés français | obligatoires |
| 4. | Le résumé en anglais et les mots clés anglais | obligatoires |
| 5. | Le résumé dans une autre langue que l'anglais ou le français (si le document est écrit dans une autre langue que l'anglais ou le français) | obligatoire |
| 6. | Le résumé de vulgarisation | facultatif |
| 7. | La table des matières, la liste des tableaux, la liste des figures ou autre | obligatoires |
| 8. | La liste des sigles et des abréviations | obligatoire |
| 9. | La dédicace | facultative |
| 10. | Les remerciements | facultatifs |
| 11. | L'avant-propos | facultatif |
| 12. | Le corps de l'ouvrage | obligatoire |
| 13. | Les index | facultatif |
| 14. | Les références bibliographiques | obligatoires |
| 15. | Les annexes | facultatifs |
| 16. | Les documents spéciaux | facultatifs |

# Université de Montréal

Faculté des arts et des sciences

Ce mémoire intitulé

**Better representation learning for TPMS**

présenté par

## Amir Raza

a été évalué par un jury composé des personnes suivantes :

*Jian-Yun Nie*
(président-rapporteur)

*Laurent Charlin*
(directeur de recherche)

*Golnoosh Farnadi*
(codirecteur)

*Jian Tang*
(membre du jury)