Learning industrial descriptions: NLP tasks for acronym expansion

Shaun **Johnson**

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Faculty of Engineering, Built Environment & IT,
Department of Computer Science, University of Pretoria, Pretoria.

A mini-Dissertation submitted to the Faculty of Science in fulfilment of the requirements for a Masters degree in Big Data Science.

Supervised by
1st Supervisor - Vukosi **Marivate**

February 19, 2024

Declaration

I, Shaun & Johnson, hereby declare the content of this dissertation to be my own work unless otherwise explicitly referenced. This dissertation is submitted in partial satisfaction of the requirements for Master degree in Big Data Science at the University of Pretoria, Pretoria. This work has not been submitted to any other university, nor for any other degree.

Signed: 

Date: 19 Feb 2024

Abstract

The human language is cryptic since words can be interpreted differently based upon the context within which they occur. The exact meaning of a particular word in its context might be trivial for humans who are generally unaware of language ambiguities. Machines, on the other hand, are required to process, transform and analyse unstructured textual information to determine the underlying meaning.

"*Acronyms*" are shorter versions of phrases and are advantageous to save time and space for both handwritten and typed out "*expansions or meanings*". The main disadvantage caused by acronyms is confusion; if misunderstood they can unknowingly cause damage, have a negative effect, or abuse the receiver. Acronyms in one context might not be appropriate for a audience in another context for the same acronym. Solving acronym disambiguation could help reduce the negative effects of using acronyms.

In this project we apply NLP technologies for a case study at a particular organisation in the Mining, Metals & Minerals (MMM) sector. The MMM organisation plant sensors' *tags (the acronyms)* are derived by domain experts from technical programmable logic controller (PLC) names into pseudo English (metallurgical) *descriptions*, these being the ground truth expansions, to describe the sensors adequately for multiple stakeholders (including non-domain experts).

There is varied human input, leading to inconsistency in initiating "*tag names (acronyms)*", and this leads to uncertainty of various degrees in trying to derive an "*accurate description from the tags (acronym expansions)*".

The aim of this research is to gauge to what extent transfer learning can be applied between similar domains using large language models. For example, Scientific document understanding could possibly explain some Mining, Metals & Minerals acronyms.

This leads us to the research question, can NLP pre-trained transformers be applied to the MMM industry for which there are low resource settings and little (or no) acronym dictionaries?

We presented a SciAD/ SDU fine-tuned transformers that can disambiguate acronyms within Scientific document understanding (SDU) context very well and is a stepping stone to being used in the Mining, Metals & Minerals (MMM) domain in future. We foresee that there is still opportunity to unlock the benefits of other pre-trained language models (PLM). We note the value that a small model could be used for the MMM domain.

Acknowledgements

I would like to thank my supervisor, Prof. Vukosi Marivate, for his enduring support, his thought provoking questions, and greatly-appreciated advice throughout my dissertation. Without his invaluable direction and depth of knowledge in this field, this work would not have been possible.

Additionally thank you to the lecturers and staff of the Master degree in Big Data Science at the University of Pretoria for all the shared knowledge and memorable class experiences and projects.
Thanks to the Data Science for Social Impact research group and staff (DSFSI) - such great memories, such great work and looking for to the diversity that will emerge in NLP tasks!

I would like to thank my class mates (some who have now become genuine friends) - we endured and with grit we did ourselves proud. Well done and all the best on your journeys going forward! Stay in touch.

I would like to thank my wife and children for supporting me through this epic journey of growth. For all their hours - often going above and beyond the call of duty - and for putting up with me behind the scenes, and providing their love and support throughout the ups and downs.

Thank you to my Heavenly Daddy whose unmerited love and favour sustained and oversupplied my needs - I am eternally grateful.

Contents

List of Figures

List of Tables

Glossary

AD acronym disambiguation. ii, 1, 2, 6, 8, 14, 42

ADI Abbreviation Definition Identification. 2, 14, 42

AI Acronym Identification. 2, 14, 42

AUROC area under the receiver operating characteristic curve. 9

BERT Bidirectional Encoder Representations from Transformers. 9, 11, 12, 14, 15, 19–22, 27–29, 31, 39, 40

Bi-LSTM bidirectional long short-term memory network. 8–11

BoW bags of words. 5, 9

BPE Byte-Pair Encoding. 11, 12, 15

CBOW Continuous Bag of Words. 10

CLM Casual Language Modeling. 12

CV cross-validation. 32, 33

EL Entity Linking. 6, 7

ELMo mbeddings from Language Models. 10, 11

EM exact match. 20, 21, 23–25

GDES gradient-disentangled embedding sharing. 22

GloVe Global Vectors for word representations. 10, 11

GLUE General Language Understanding Evaluation. 14, 22

IE information extraction. 6

IOB inside-outside-beginning. 9

KB knowledge-based. 8

KD knowledge distillation. 14, 22

LF long-form. 19

LSA latent semantic analysis. 6

LSTM long short-term memory. 3, 10, 12, 18, 19, 26, 27, 35, 40

MLM Masked Language Modelling. 11– 13, 21

MMM Mining, Metals & Minerals. ii, vi, viii, 1, 2, 16, 17, 23, 25, 28– 30, 39– 42

NER Named Entity Recognition. 9, 14

NLM National Library of Medicine. 10

NLP Natural language processing. ii, iii, 1– 3, 10, 19, 22, 39

NLTK Natural Language Toolkit. 17

NSP Next Sentence Prediction. 11, 13, 21

OOV out of vocabulary. 11

PHI Protected Health Information. 9

PLC programmable logic controller. ii, 1

PLM pre-trained language models. ii, 12, 14, 41

RE Relation Extraction. 9

RNN random neural networks. 10, 12

RTD Replaced Token Detection. 13

RTS Random Token Substitution. 13

SDU Scientific document understanding. ii, vi, 2, 8, 16, 22, 27, 28, 32, 33, 35, 37, 39–42

SF short-form. 19

SLM Swapped Language Modeling. 13

SOP Sentence Order Prediction. 13

SOTA state-of-the-art. 2

SQuAD Stanford Question Answering Dataset. 20, 22

SS Semantic Similarity. 9

SVM support vector machines. 6, 8

TAPT task-adaptive pretraining. 15

TF TensorFlow. 20

TI Textual Inference. 9

TLM Translation Language Modeling. 13

UMLS United Medical Language System. 10

URL uniform resource locator. 16, 17

WSD word sense disambiguation. 3, 5, 6, 8, 10, 14

Chapter 1

Introduction

Natural language processing (NLP) has grown exponentially as a discipline in the last decade and has found use within word sense disambiguation in the biomedical and clinical domains, and is moving to other domains and language tasks of scientific document understanding. These domains share a common task of acronym disambiguation.

Background to acronym disambiguation (AD): "*Acronyms*" are shorter versions of phrases and are advantageous to save time and save space for both handwritten or typed out "*expansions or meanings*". Acronyms can be easier to "read" or used for quicker communication. From a word count perspective, acronyms can optimise a manuscript or technical document by reducing the number of "repeated phrases". New acronyms for novel techniques can get "catchy" names which are more memorable. The main disadvantage caused by acronyms is confusion, and if misunderstood can unknowingly cause damage, have a negative effect, or abuse the receiver. Acronyms in one context might not be appropriate for an audience in another context for the same acronym. Solving AD could help reduce the negative effects of using acronyms.

In this project we apply NLP technologies for a case study at a particular organisation in the Mining, Metals & Minerals (MMM) sector. MMM organisation plant sensors' *tags (the acronyms)* are derived by domain experts from technical programmable logic controller (PLC) names into pseudo English (metallurgical) *descriptions, this being the ground truth expansion*, to describe the sensors adequately for multiple stakeholders (including non-domain experts).

There is varied human input, leading to inconsistency in initiating "*tag names (acronyms)*", and this leads to uncertainty of various degrees in trying to derive an "*accurate description from the tags (acronym expansions)*".

1.1 Problem Statement

Sensor instrument tags are (mostly) ordered strings comprising unique IDs and several acronyms which depict sensor classification, name-entity, and/or geo-location information.

The acronyms which make up the string are typically supposed to be a few characters in length. Certain acronyms that represent the main name-entity are to follow an industrial standard and in some cases it has not been adhered to; strings tend to be a combination of various types of acronyms. There are no clear delimiters (for example, white-space, character case, or fixed lengths) to distinguish combined acronyms and acronym tokens.

Data users might not clearly understand the sensor tag description which could impact any data analytic efforts, since an incorrect sensor tag may unknowingly have been selected for a study or report. The sensor tag description is arguably the first opportunity to give a clear generic picture of the sensors (meaning context to the feature) without involving a domain expert or hindering self-service (big) data analytics tasks.

Note that each sensor instrument tag can be written in an expanded and more understandable pseudo English form which we will call the sensor description.

1.1.1 Research Aims

The aim of the this research is to gauge to what extent transfer learning can be applied between similar domains using large language models. For example, Scientific document understanding (which is technical and there are public corpora like academic papers) could possibly explain some Mining, Metals & Minerals acronyms (where there are low resources or sparsely defined dictionaries). This leads us to the research questions.

1.1.2 Research Questions

Not all sensor tag descriptions AD expansions) are adequately described or of a good "data quality" from a generic perspective - for example, do sensor tag descriptions have the elementary phrases in the context to exactly or nearly match the sensor tag names (comprising one or several acronyms)? So, the first natural task would be to sense check what acronym expansions exist for the MMM domain and exist in the scientific and/or in the general language domain. Therefore the overall research question is, can NLP pre-trained transformers be applied to the MMM industry for which there are low resource settings and little (or no) acronym dictionaries?

- What performance can state-of-the-art (SOTA) pre-trained transformer models deliver when applied to the MMM domain data in a zero-shot application?
- Is there more potential to disambiguate acronyms using generalized autoregressive pre-training methods?
- How do smaller main stream autoencoding based models perform?

1.1.3 Scope and limitations

This work is focused on AD, which is a downstream task to Acronym Identification (AI) or Abbreviation Definition Identification (ADI).

Anticipated model limitations are expected to occur from the base pretrained models themselves, in terms of the data the models were trained on - so there might be bias. For example, models trained on English will be used (instead of multilingual language models).

Sourcing public datasets that contain the necessary features of acronyms, acronym expansions and context example features, but are not of high quality will impact this work. Any dataset might itself not be balanced due to the nature of limited access to domain-specific knowledge. We are, however, grateful for the good quality public datasets out there, exepcially when considering the effort and resources these take.

Choices made regarding data preprocessing and/or modeling parameters can skew the results - the results will thus have a dependency on the datasets used, as well as the model parameters chosen.

Upfront no data curation will be done; instead the dataset sourced will be preprocessed such that model size limits are not a limiting factor. The textual context will be simplified to reduce noise (such as replacing quantities with a single numeric string, removal of special cases strings, and only allowing alphabetical characters in the training data).

Insufficient training data or data that does not fully represent the real-world or edge cases could result in NLP hallucinations if generative AI is pursued. For this work, we are not "generating or inferencing" outputs, but rather performing an "understanding" task.

1.2 Summary

The outline of this dissertation is as follows.

In Chapter 2 we present a literature review comprising two sections - firstly, a background of word sense disambiguation (WSD), and WSD related to acronym and context investigations in the biomedical and/or clinical domains. Then, secondly, an overview of machine learning technologies or methods on cutting edge embeddings and self-supervised learning methods is presented. A summary of acronym tasks are defined, and thereafter related work in the scientific domain discussed.
In Chapter 3 we suggest a methodology for experiments. The baseline end-to-end memory architecture which is LSTM-based is described. Individual transformers models or an ensemble setup is discussed, and how they experiments will relate the the research questions.
In Chapter 4 experiment results are reported. Baseline results and individual transformer evaluation metrics are recorded and described. We further assess the predictability, computability and stability of the work done.
In Chapter 5 we discuss the results in several components, namely the key findings, interpreting the findings and a discussion of the limits of the study.
In Chapter 6 we conclude and formulate recommendations for future work.
Appendix A has more detailed tables of experiment results, Appendix B gives samples of predictions, and Appendix C documents the modelling process in terms of training and validation

monitoring, and code requirements for python packages, and a brief hardware description relating to model experiments done within this work.

Chapter 2

Literature Review

2.1 Word Sense Disambiguation

2.1.1 Background

A good summary of word sense disambiguation may be found in [Navigli, 2009]. The human language is cryptic since words can be interpreted differently based upon the context they occur. The exact meaning of a particular word in its context might be trivial for humans who are generally unaware of language ambiguities. Machines on the other hand are required to process, transform and analyse unstructured textual information to determine the underlying meaning. Word sense disambiguation is the computational identification of word meanings (i.e. the assigning of appropriate senses) based on the context in which the words exists. WSD is extremely dependant on knowledge to infer the most sensible meaning of words. Both humans and machines can only extract word meanings if knowledge sources exist. Knowledge sources such as unstructured collection of texts (corpora) or structured resources (such as dictionaries, semantic networks, ontologies, etc) are available. Note, however, that it is well known that manual creation and upkeep of such knowledge resources is expensive and time consuming.

The process of identifying the word sense using WSD normally begins with the usual preprocessing steps, e.g. tokenization, part-of-speech tagging, lemmatization, chunking and parsing. A set of features is then used to represent the text (which may include information derived from these preprocessing steps). Features broadly fall into the following categories:

- Local features which focus on investigating words situated close to the target word, for example their part-of-speech tags and position relative to the target word.
- Topical features, often given as bags of words (BoW), which represent more general contexts, describing a window of words up to a whole paragraph.
- Syntactic features, which may be outside the local context, that represent syntactic cues between the target word and the other words in the same sentence.

- Semantic features which describe previously established senses of words in context.

WSD may be supervised or unsupervised. In supervised WSD, machine-learning techniques use labeled training sets (i.e. examples encoded in terms of a set of features) together with the applicable sense label. Methodologies that have historically fallen within this category include decision lists, decision trees, naive Bayes, neural networks, support vector machines (SVM) and various ensemble methods such as majority voting, probability mixture, rank-based combination and AdaBoost.

Unsupervised WSD, on the other hand, is based on unlabled corpora. Methodologies falling within this category include context clustering - which incorporates cosine similarity and latent semantic analysis (LSA), word clustering and cocurrence graphs.

The line between supervised and unsupervised learning is hazy, and some methods are minimally supervised or semisupervised and can learn sense classifiers from annotated data. One such method is bootstrapping which aims to solve the lack of annotated data and the data sparsity problem, which generally uses either a cotraining or self-training approach. Here one needs to choose a set of heuristics to follow, for example in Yarowsky's approach, the heuristics employed are that of one sense per discourse (i.e. within a particular discourse, a word is consistently referred to in the same sense) and of one sense per collocation (i.e. relative distance, order and syntactic relationship of nearby words consistently contribute to determining the sense of the word).

The biomedical field is an example of this, and much work has been done within this subdomain in recent years. In the task of information extraction (IE) and solving the ambiguities of domain specific concepts the specific task of acronym expansion can be classified as a disambiguation problem. We therefore review several pertinent studies to glean useful methodologies.

2.1.2 Acronym disambiguation

Acronym (or abbreviation) sense is a subset or special case of WSD. An area ripe for the study of accronym disambiguation is the clinical domain, since there are only a few available acronym/ abbreviation datasets for the clinical domain, and clinical notes are informal in practice and implies the structure and formatting are fine between clinicians (or domain-experts) and not intended for re-use (or opportunities for automation).

[Li et al., 2018] did some important acronym disambiguation (AD) work for Enterprises by adopting previous work; previous work namely:

- Generic rules or text patterns methods: Enterprises using generic rules are not adequate since it assumes acronym and acronym expansions co-exist in the same context (whereas for Enterprises they are typically found apart).
- Entity Linking (EL) or special case of AD methods: EL relies on public meanings, however for Enterprises, the public meanings generally do not correlate. The Entity Linking algorithms either:

  - Calculate the cosine similarity between the acronym and the acronym's document; or
  - Calculate the topical coherence between the acronym and other relevant documents based on overlaps or links; or
  - Calculate a hybrid of the two measures.

[Li et al., 2018] took the strengths of generic rules and Entity Linking and proposed an acronym mining stage that is done offline and requires the input Enterprises corpus to create an on-demand reference dictionary. The high-quality mining process quantifies the raw data on

- candidate generation (to find the acronym expansions, firstly from the Enterprise corpus, and thereafter from the public domain, and categorises candidates based on popularity scores to infer the genuine meaning),
- candidate deduplication (to keep the best or exact expansion to represent a group of similar and varying expansions), and
- context harvesting (to store the context as metadata, additionally the best context window seemed to be **30 tokens** or words surrounding the acronym).

The candidate generation techniques could yield 94.5% valid labels from the 2000 random samples selected from the 17285 mined acronyms and expansions.

The three fit-for-purpose models are candidate ranking, confidence estimation and final selection. The candidate ranking model makes predictions based on likelihood via a boosted tree algorithm. The confidence estimation model, which is also a boosted tree algorithm, is used as a classification model to decide whether to trust the best ranked answer or not; and a confidence threshold is used to block misleading expansions. The final selection model helps discern the ranks of the top results, meaning popular public expansions with a low confidence estimation will move down the ranks of the reported results of the most real meaning expansion. The combination of the models are able to outperform state-of-the-art Entity Linking (EL) systems: Wikifier and AIDA on Microsoft Office365 documents (that are not in the public domain).

In [Charbonnier and Wartena, 2018] a set of 19,954 examples of 4,365 ambiguous acronyms was constructed using image captions as found in scientific papers, together with their contextually correct definitions from different domains. The context comprised either the image caption, the acronym it was taken from or the caption expanded with sentences referring to the image. (If more context was required due to less than 15 tokens being in the caption, the referring sentence was added to the context, and then left and right neighbour sentences until it had a total of 35 tokens. The unsupervised classifier was defined as follows. The predicted definition of acronym a given context C is

$$\text{pred}(a, C) = \max_{D \in \mathcal{D}} \cos\left(cv(D), cv(C)\right)$$

where, for $t = s_0, s_1, ...s_n$ a sequence of tokens, the context vector $cv(t)$ can be written in terms of a sum involving the context vector (i.e. word embedding) of the tokens s_i,

$$cv(t) = \frac{1}{N} \sum_{i=0}^{n} idf(s_i) \cdot cv(s_i).$$

It was found that all methods using the acronym's context performed better than the majority classifier, and that adding more context for short captions resulted in small improvements. Using simple word overlap gave better results than using pre-trained vectors. The unsupervised classifier used (via a form of cosine similarity) in tandem with corpus specific word embeddings, performed better than all of the other variants. For cases where the cosine similarity was at least 1, an accuracy of over 0.95 could be achieved. They envisaged that follow-up research included learning text representations with neural networks, and thereafter learning text similarity as well.

In [Zhong et al., 2021], they tackle acronym disambiguation (AD) in Scientific document understanding (SDU) through a **h**ierarchical **d**ual-path **BERT** (*hdBERT*) method to capture "the general fine-grained and high-level specific representations for acronym disambiguation".

2.1.3 WSD in a biomedical and/or clinical domain

In [Wu et al., 2017], an open-source practical solution was developed to generate corpus-specific "sense inventories". Their **c**linical **a**bbreviation **r**ecognition and **d**isambiguation (CARD) solution is based on a framework comprising three steps: (1) machine learning algorithm (support vector machines (SVM)) and using available clinical abbreviation lists to recognise abbreviations, (2) sense detection using clustering techniques to obtain centroid sentences, and (3) sense disambiguation step based on robust profile-based word sense disambiguation methods from [Xu et al., 2012]. The key challenges were no complete clinical abbreviation lists existed and clinical abbreviation are ambiguous. Clinical abbreviations are typically invented by health care providers in an adhoc manner, and thus fall under WSD. [Wu et al., 2017] learnt that WSD supervised machine learning yields great results but were potentially not worth it due to the heavy need to annotate ambiguous abbreviation training sets. So they used profile-based WSD to overcome the cost of annotation.

A single bidirectional long short-term memory network (Bi-LSTM) outperformed supervised WSD that used a PageRank algorithm on occurrence graphs and was in the same range as unsupervised knowledge-based (KB) models setups. Key learnings of Bi-LSTM methods were the selection of direction impact accuracy and most for our purpose pre-trained embedding have an advantage for the same WSD task.

Past relevant work: A significant amount of work relevant to our research objectives has been carried out by different research groups, and we highlight some contributions below (refer to Section 2.2 to understand technologies that these contributions refer to).

- The effects of injecting various types of noise into a dataset on Named Entity Recognition (NER), Relation Extraction (RE), Textual Inference (TI) and Semantic Similarity (SS) was investigated in [Moradi et al., 2021]. On average, considering on an individual basis each of the character-level noise types (deletion, insertion, letter case changing, commonly misspelled words, repetition, replacement and swapping) and word-level noise types (replacement with abbreviation, abbreviation expansion, deletion negation, word order, repetition, replacement with synonym, singular / plural verb and verb tense) resulted in 6-15% reduction in performance as compared with the original test set. (Here, on average, ClinicalXLNet performed better than ClinicalBERT, which performed better than ClinicalELMo.)

- [Oh et al., 2022] aimed to determine which of BERT, RoBERTa and XLNET would best perform in the task of Protected Health Information (PHI) identification. It was found that XLNET performed best indicating a greater propensity for understanding the context. XLNET also showed superior performance as a pretraining model; this was attributed to word embedding being well constructed using the two-stream self-attention method. Note that in this work, quite a bit of preprocessing was involved in the form of tokenisation, inside-outside-beginning (IOB) tagging, and word piece tokenisation, prior to being input into BERT, RoBERTa or XLNET.

- [Huang et al., 2019] found that for the task of predicting mechanical ventilation using the area under the receiver operating characteristic curve (AUROC) score to perform comparisons, while BERT outperformed XLNET, Clinical CLNET outperformed ClinicalBERT. This demonstrates firstly the importance of pretraining on domain-specific corpus, and secondly that Clinical XLNET has the advantage due to the usage of sequential modeling of the temporal dimension of notes since time order matters in clinical prognoses.

- [Huang et al., 2020] compared the performance of BERT to ClinicalBERT in firstly language modelling and next sentence prediction, and thereafter in predicting hospital readmission. A massive improvement from around 0.5 to 0.9 (with standard deviation of 0.002) using ClinicalBERT was seen for the former. Similarly for the latter, ClinicalBERT outperformed BoW, Bi-LSTM, and BERT in accurately predicting 30-day readmission using discharge summaries.

2.2 Technologies

2.2.1 Cutting edge embeddings

While generic domains enjoy copious amounts of information, and numerous robust available methodologies, dictionaries and ontologies, often-times our problem statement relates to a specific sub-domain. In these cases ones needs to be able to find ways of making use of the generic-domain learnings, and overcome various obstacles such as comparatively low data and limited dictionaries to name a few.

In [G.K. et al., 2003], they had 25 million Mayo clinic electronic clinical notes, and they needed to create a suitable clinical corpus as well as manual sense annotations (they were applying a supervised machine learning methodology). They were able to compare their (clinical domain) results with those in the biomedical literature space which has the benefit of NLP tools developed at the National Library of Medicine (NLM) as well as the United Medical Language System (UMLS) ontology. The latter has been reported to have 7400 ambiguous words which has made it an ideal candidate for investigating WSD methodologies.

In order to perform this analysis on the unstructured textual data obtained from the clinical domain, they employed a work-flow that involved identifying the top 50 relevant ambiguities in the large clinical note corpus (the MetaMap tool and the United Medical Language System (UMLS) ontology were useful here in selecting these ambiguities), and applying and evaluating a (then) state-of the art WSD machine learning algorithm to small datasets in which up to 120 sense-labeled instances of each ambiguity were identified (from both the clinical and biomedical domains). Thereafter the most productive features were pin-pointed patterns in the text within a vicinity of the ambiguous sense.

More recently, a survey of 70 relevant biomedical word embedding papers between January 2016 to September 2020 was done on cutting edge NLP models [Chiu and Baker, 2020]. Cutting edge embeddings include

- Global Vectors for word representations (GloVe), here a word co-occurrence matrix is generated where each word in a particular sentence is said to be in the context of each of the others in that sentence). GloVe learns neural embedding of word co-occurrence frequencies at sentence level.
- Word2vec, which makes use of tools like Continuous Bag of Words (CBOW) and skip-gram which enable neural embeddings to be learnt through a neural network.
- FastText, which is an extension of Word2vec but allows for both word and character (morphological)-level information which is of particular use in domains with complicated words like the biomedical field. By utilising sub-word information, FastText can provide embeddings for unseen words. This is because even if a testing word is unseen during training, its embedding can be obtained using the sum of its character n-grams.
- ELMo consisting of multiple layers of Bi-LSTMs (Bidirectional Long Short-Term Memory) where character-level embeddings encode both contextual and sub-word information. LSTMs which, according to its creator Jürgen Schmidhuber, is the most cited neural network of the 20^{th} century, consist of an input gate, an output gate and a forget gate. LSTM units can allow gradients to flow unchanged which helps to solve the vanishing gradient problem that arises in training vanilla random neural networks (RNN) using back-propagation. Since ELMo uses Bi-LSTM in training, its language model understands the next word as well as the previous word in sentences. Each word (k)'s embedding is given by the weighted sum

$$ELMo_k^{task} = \gamma_k \cdot \left(s_0^{task} \cdot X_k + s_1^{task} \cdot H_{1,k} + s_2^{task} \cdot H_{2,k}\right),$$

where s_i are the weights, X_k the character-level embeddings, $H_{i,k}$ the hidden states of the Bi-LSTMs, and γ_k a task-specific scaling factor allowing ELMo's representation to be fine-tuned based on its downstream tasks such as sentiment analysis and document classification. The weighting factors s_i and γ_k are updated during task-specific model training and frozen (and concatenated with input representation) when ELMo is used within a task.

- Bidirectional Encoder Representations from Transformers (BERT) considers three embedding types - namely, token embeddings (as learned for the particular words from the WordPiece vocabulary), sentence embeddings (BERT predicts which of two sentences comes first, and which of the sentences a particular word would belong) and position embeddings where a particular word would be positioned within a sentence. The first step in BERT's functionality is the step of input embeddings, where input text is embedded into vectors. Under Masked Language Modelling (MLM), 15% of words in each sequence are randomly masked, and it is the model's job to try to predict these masked words based on the context of the unmasked words. Note that there is therefore no bias towards either the left-to-right or right-to-left direction, so that words of all positions have èqual opportunity'. Under Next Sentence Prediction (NSP), BERT tries to predict whether 2 sentences are consecutive. Thereafter a Self-Attention layer maps these input embeddings to an output vector by taking the dot product of a query Q with all keys K, dividing by the square-root of the dimension of K. Taking the softmax of these results in a set of weights for keys V, where (K, V) is a set of key-value pairs. In other words, the self-attention is given as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{Q \cdot K}{\sqrt{d_k}}\right) V$$

  Whilst both the quickly trainable GloVe and Word2vec are morphological (i.e. character) information-agnostic and contextual information-agnostic (and are unable to handle out of vocabulary (OOV) words), FastText encodes morphological information but remains contextual information-agnostic. Both ELMo and BERT leverage both morphological information and contextual information. ELMo experiences some locality bias, where models focus on words closer to target-trained words since in ELMo a text sequence is analysed either from left-to-right or combined left-to-right and right-to-left; this bias is eradicated in the BERT model. ELMo and BERT tend to outperform Word2vec and GloVe, and are able to achieve cutting-edge performance.

- RoBERTa (A Robustly optimised BERT approach) is an approach that has found that certain design decisions in terms of pretraining BERT models could substantially improve performance [Liu et al., 2019b]. In particular this strong improvement in performance came from training the model longer, having bigger batches over more data, removing the next sentence prediction objective, training on longer sequences, and dynamically changing the masking pattern that was applied to the training data.

- CharBERT is a character-aware pre-trained language model [Ma et al., 2020b]. It has a slight edge over BERT when dealing with subwords by including character-level information. The character-level extends the advantages of Byte-Pair Encoding (BPE) in OOV

cases. There is a lack of robustness when using BPE WordPiece due to incomplete modeling since fine-grained character information and whole word representation may not be incorporated in the subword representation; and secondly minor typos can have significant impact on the BPE token leading to inaccurate and incomplete representations. CharBERT, which has been evaluated on 8 benchmarks (showing a significant performance improvement compared with BERT and RoBERTa baselines) and tested using three character attack test sets on three types of tasks (showing a large robustness improvement), is able to enrich the word representation in pre-trained language models (PLM) since they incorporate features at an addition level of a word in the model.

- Transformer-XL, which forms the basis of XLNET, addressed issues related to (RNNs and the associated LSTM networks which are difficult to optimise due to gradient vanishing and explosion, as well as issues relating to predefined context length such as no information flow across segments which leads to models not being able to capture longer-term dependencies beyond the fixed context length [Dai et al., 2019]. It did this by reusing hidden states obtained in the previous segment rather than computing them from scratch, so that modeling of long-term dependency is possible and the problem of context fragmentation is solved. Also, relative position encoding was introduced instead of relative ones. Transformer-XL obtained strong results at both character- and word- level language modeling (being the first self-attention model to see substantially better results than (RNNs for both) and could achieve relatively coherent results using a fraction of input data.

2.2.2 Self-Supervised Learning (SSL)

BERT and models like it are modeled using Self-Supervised Learning, where instead of the human manually labeling the training data set, these labels are automatically generated by the model, based on data attributes and the pretraining task definition [Ma et al., 2020a]. Language representations are learned using a pretraining loss function (that we try to minimise) which, in general, is a sum of m pretraining tasks multiplied piece-wise by m weights,

$$L_{SSL} = \lambda_1 L_{PT-1} + \lambda_2 L_{PT-2} + ... + \lambda_m L_{PT-m}.$$

Below we take a look at some of the various forms that these pretraining tasks can come in:

- In Casual Language Modeling (CLM), one tries to predict the next word based on the context, in particular, based on the words to the left of the given word, or to the right of it (i.e. this falls under unidirectional Language Modeling). The CLM loss is given by

$$L_{CLM}^{(x)} = -\frac{1}{|x|}\sum_{i=1}^{|x|} \log P(x_i|x_{<i})$$

  where $x_{<i} = x_1, x_2, x_3, ..., x_{i-1}$ and $|x|$ is the number of tokens in the sequence.

- In Masked Language Modelling (MLM), the pretraining task which RoBERTa uses and one of the two pretraining tasks which BERT uses (in BERT, the authors masked 15% of

tokens). The MLM loss is then given by

$$L_{MLM}^{(x)} = -\frac{1}{|M_x|}\sum_{x\in M_x}\log P(x_i|x_{\setminus M_x})$$

where M_x is the set of masked token positions in x and $x_{\setminus M_x}$ represents the masked version of x. There are various of extensions of MLM, two of which we describe below.

- Translation Language Modeling (TLM) , also called cross-lingual MLM (XMLM), uses parallel data in cross-lingual pretraining. The input is a pair of sentences (x, y) where say y is a translation of x, and, similar to MLM, both sentences are randomly masked. The TLM loss function is then given by

$$L_{MLM}^{(x,y)} = -\frac{1}{|M_x|}\sum_{x\in M_x}\log P(x_i|x_{\setminus M_x,\setminus M_y}) - \frac{1}{|M_y|}\sum_{x\in M_y}\log P(x_i|x_{\setminus M_x,\setminus M_y}).$$

- Swapped Language Modeling (SLM) works the same as MLM but instead corrupts the sequence with random tokens from the vocabulary with probability 15% (this overcomes the discrepancy between the pretraining and fine-tuning stages which MLM experiences). The SLM loss function is given by

$$L_{SLM}^{(x)} = -\frac{1}{|R_x|}\sum_{x\in R_x}\log P(x_i|x_{\setminus R_x}).$$

- The Replaced Token Detection (RTD) is made use of by ELECTRA. Here, output tokens are obtained from a generator model trained using the MLM objective, and then uses these to corrupt the sentence. RTD is then a token-level binary classification task which aims to predict whether a token is predicted or not. The RTD loss is defined as

$$L_{RTD}^{(x)} = -\frac{1}{|\hat{x}|}\sum_{i=1}^{|\hat{x}|}\log P(d|\hat{x}_i).$$

where $d \in \{0, 1\}$ depending on whether the token is replaced or not.

Various model are based on the same concept as RTD, for for example Sentence Order Prediction (SOP) and Next Sentence Prediction (NSP). We briefly describe another of these, Random Token Substitution, below.

- Random Token Substitution (RTS) is sample efficientlike Replaced Token Detection (RTD) but does not need as separate generator to corrupt the input sequence, and instead uses randomly substituted 15% of the tokens from the vocabulary. The loss function is similarly given by

$$L_{RTS}^{(x)} = -\frac{1}{|\hat{x}|}\sum_{i=1}^{|\hat{x}|}\log P(d|\hat{x}_i).$$

It has been shown that the RoBERTa model trained using RTS has the same performance as the RoBERTa model trained using MLM, but requiring less training time.

2.2.2.1 TinyBERT

[Kovaleva et al., 2019] revealed that disabling certain self-attention heads were possible and lead to no drop in model accuracy and found fine-tuned BERTs suffered from over-parametrisation. Smaller models that achieved comparable performances to the BERTs are possible and with task-specific distillation methods.
Such task-specific distillation have led to competitive fine-tuned tinyBERTs. [Jiao et al., 2019] looked for the means to reduce the size of large parameterised PLM as well as improve the inference time experienced by edge devices like mobile phones. They focused on knowledge distillation (KD) technique for transformers, rather than the most common methods of weight pruning or quantisation. The KD philosophy that was proposed used a two-stage learning framework to facilitate the teacher BERT transferring knowledge to student tinyBERT.
tinyBERT$_4$ is 7.5 times smaller and capable of 9,4 times faster inference while maintaining 96.8% GLUE benchmark performance of the teacher BERT$_{BASE}$.

2.2.3 Acronym tasks

The acronym tasks we are interested in are those of [Church and Liu, 2021]

- Abbreviation Definition Identification (ADI), which is the realistic task in most practical application, takes one or more texts as input and determines pairs of short forms (SF - abbreviation / acronym) and long forms (LF - expansion of acronym) as defined in input texts, with no restriction on the length of the input texts.
- Acronym Identification (AI), which is similar to NER, and takes a text as input and provided each input word with a particular tag, e.g. a B (begin), I (Inside) or O (Outside) followed by -long if it is a LF, or -short if it is a SF.
- acronym disambiguation (AD), which takes a sentence with an ambiguous SF as input and outputs its appropriate LF; this links to WSD as has been discussed above.

2.2.4 Related work

2.2.4.1 Hierarchical Dual-path BERT (hdBERT)

In subdomains it becomes useful to use domain specific versions of BERT. For example, when comparing the performance of BERT and BioBERT (trained from general domain text and bio-medical corpora), BioBERT performed noticeably better (2-5 points in F-score on average). SciBERT, which is based on fine-tuning of pre-trained BERT using scientific publications from both computer science and biomedical domains, has in turn outperformed BioBERT. Further improvements to SciBERT have been sought in the form of constructing a science specific vocabulary rather than BERT's default dictionary (unlike in BioBERT) among other things. FinBERT also exists, obtained by finely tuning generic BERT embeddings using financial data from various online financial sources. In a further development, XLNet has been proposed to help solve the

long-text dependency by replacing consecutive sentence modelling with modelling of permutations of sentences, and appears to give better results than BioBERT and ClinicalBERT.

In the scientifc domain, a further construction comes in the form of combining models from the above; for example, hdBERT comprises two context-based pre-trained models (RoBERTa and SciBERT) integrating into a multiple layer perceptron (or a neural networks) to predict outputs [Zhong et al., 2021]. Hierarchical Dual-path BERT (hdBERT) incorporates the general domain and fine-grain (BPE) representations using RoBERTa and the (scientific) domain specific knowledge using SciBERT as inputs to a binary classification gate. The loss function is defined as

$$L(\sigma) = -\sum_{D}(y\log(p) + (1-y)\log(1-p) + \lambda\|\sigma\|_2^2$$

where ground truth is y, hdBERT parameters are σ, regulariser parameter is λ and trainset D.

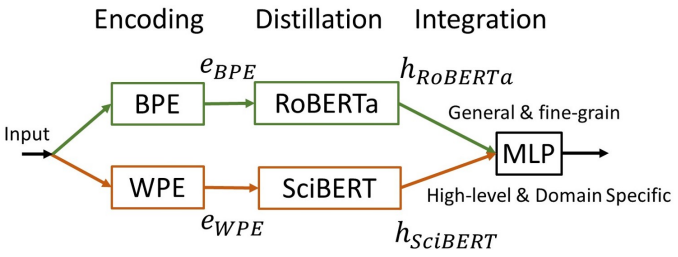


FIGURE 2.1: Conceptual hdBERT model comprising RoBERTa and SciBERT outputs from a multiple layer perceptron network and sigmoid unit.

In Figure. 2.1 hdBERT illustrates data flow where the input sentence sample comprising acronym and expanded long form is split to obtain BPE, e_{BPE} and SciBERT encoding, e_{WPE}. In parallel the models distill the information and produce RoBERTa output representations, $h_{RoBERTa}$ and SciBERT out representations $h_{SciBERT}$. The model representations are integrated into the multiple layer perceptron network and sigmoid unit.

2.2.4.2 Acronym Disambiguation with Multiple Training Strategies

Work by [Pan et al., 2021] leveraged pre-trained BERT family models as binary classification models and implemented several training strategies to substantially improve a BERT-based model for scientific document understanding acronym disambiguation.

The SciBERT was re-trained with " task-adaptive pretraining (TAPT) as the new pretrained model. The training steps applied were "Dynamic Negative Sample Selection" (to have a more balanced training dataset), and "Adversarial Training" (by adding perturbations to the embedding layer for a more generalised model to handle unseen data). And incorporating a training step involving "Pseudo-Labeling" a synthetic training dataset further improved the model to near human level performance.

The binary classification model inputs a sentence containing an acronym and a expansion, and the model outputs a prediction score of the inputted expansion.

Chapter 3

Methodology

We will treat the acronym disambiguation task as a closed-domain question answering problem. The models will learn how to discern the acronym expansions in context based on having learnt simple question contain the short form acronyms from training context.

3.1 Data

An open datasets was identified and used to develop the potential MMM acronym disambiguation solution of fine-tuned SDU models. And a zero-shot approach of the fine-tuned SDU models was applied to a sample MMM dataset.

3.1.1 Open dataset: SciAD for SDU

[Veyseh et al., 2020] provide an shared task dataset called SciAD for acronym disambiguation. SciAD comprises acronyms used for Scientific document understanding (SDU). The scientific domain acronyms were sourced from 6 786 English research papers comprises 62 441 sentences. The dataset contains the identified acronym and the correct expansion of the acronym; and the dictionary of 732 ambiguous acronyms has been collected and contains acronym potential long forms (or expansions).
The time and efforts of acronym identification and providing the correct expansion of the acronym in the open or public SciAD dataset is highly appreciated for this work.

Open dataset:
Data was preprocessed (cleaned and transformed) for model encoding. Identified in the SciAD data were English words, none English words, scientific jargon, numerical text is various formats as equations or different formats of a quantities; as well as punctuation and uniform resource locator (URL) links.

The SciAD data for this study was simplified in the following way.

- Quantity text was detected by a regular expression and replaced with a generic '123' string. For example, 321.09e-5 becomes 123. For this work MMM data would not have complicated equations nor numerical formats within the context.
- None alphabet characters were replaced by underscores and meaning accent characters were constant. And if the entire word comprised none alphabet characters were removed from the context. The test example, *àèìòùaeiou* would become _____*aeiou*, and the test example, '恭喜发财' ('May you be happy and prosperous[1]') would be removed. For this work MMM data typically would not have accent or special characters within the context.
- NLTK Tokenizer Package used to divide strings into lists of substrings. Lower case was applied and isAlpha() method removed URL for example "http://helloworld" and consequently removed any hyphenated words. For this work MMM data typically URLs are not recorded, but the hyphenated words could exist.
- special cases: " - " or "o - d " as empty string; and " 's" to "s" were cleaned up for general purposes.
- the context length was capped to 32 tokens (and based upon acronym position results in Figure 4.12).

3.1.2 Domain dataset: MMM data

A sample set of 50 examples was selected from the MMM data, and the ground truth or expansion was annotated.

id	### question	### context	### expansion
id1	### What is WH?	### Portable Water Reservoir Inlet 2 Flow warning high limit	### warning high
id2	### What is WL?	### Portable Water Reservoir Inlet 2 Flow warning low limit	### warning low
id3	### What is PV?	### Portable Water Reservoir Inlet 2 Flow process value	### process value
id45	### What is PD?	### Cyclone 1 differential pressure Input warning high	### differential pressure
id46	### What is PD?	### Cyclone 1 differential pressure Input warning low	### differential pressure
id47	### What is PD?	### Cyclone 1 differential pressure Output value	### differential pressure
id48	### What is SEC?	### Specific Energy Consumption Input warning high	### Specific Energy Consumption
id49	### What is SEC?	### Specific Energy Consumption Input warning low	### Specific Energy Consumption
id50	### What is SEC?	### Specific Energy Consumption Input alarm high	### Specific Energy Consumption

FIGURE 3.1: Subset of the Sample MMM data.
9 examples are shown from the 50 annotated examples.

In Figure. 3.1 a clipping of nine MMM data examples. We observe in the nine examples of the sample set of 50 examples. The first three cases (id1, id2, id3) are have the acronyms expansions near the end of the context (or MMM descriptions); the next three cases (id45, id46, id47) the expansion are in the middle and the acronym characters are swapped (and is an domain specific rule for "differential" measures); and the last three cases (id48, id49, id50) the expansion are at the beginning and the acronym is three characters long.

[1] Google Translate

3.2 Baseline

A baseline MemN2N model was established using modern memory techniques, and state-of-the-art long short-term memory (LSTM) algorithm.

3.2.1 End-to-End Memory Architecture

[Sukhbaatar et al., 2015] presented an end-to-end memory (recurrent neural network) architecture, abbreviated as MemN2N, with a recurrent attention model capable of handling long external memory and this approach was shown to be fit for question and answering tasks and more. The emphasis on the end-to-end implied less supervised training requirements from the input-output pairs when compared to [Weston et al., 2014] Memory Network architecture.

[Sukhbaatar et al., 2015] approach has two memory representations, namely an input and an output memory in order to make the final answer prediction.

Input memory is the inputs embedded into an input embedding matrix, I of d dimensions by the vocabulary size, v. And similarly the questions are embedded into question embedding matrix, Q. The relationship between the inputs and questions are matched by dot product and followed by a softmax to obtain a probability vector, P. This probability vector thus gives the best matches of the question to the inputs and the entire context of the input is taken into consideration before finding the answer to the question.

Output memory is the ground truth answers embedded into an embedding matrix, O. And a response vector, R is computed by the sum over each answer and the corresponding probability from the probability vector P.

Predictions are generated by a softmax of the sum of the response vector, R and the question embedding matrix, Q.

Algorithm 1 A long short-term memory end-to-end memory model for question and answering task.

Import packages and Initialise all variables.
Import pre-processed data and split into train and validation datasets.
Data transformation:
    Build tensors: inputs, questions, answers.
Compile Model (optimiser, loss, metrics):
    Embed inputs, questions and outputs into a sequence of vectors or encoders.
    Relate the similarity between input and questions by dot product.
    Add the related and questions and then concatenate with the responses.
    Prepare tensor of LSTM dimension.
Fit LSTM Model ([inputs_train, questions_train],answers_train)
    Minimise valuation loss and Save best only.
    Early stop with patience of 5.
    Save model files. Garbage collection.
Assess Model (model, validation_data):
    Generate predictions.
    Compute Accuracy, Precision, F1 score, and Cohen Kappa score

In Algorithm 1 the baseline modelling workflow of the LSTM end-to-end memory model (MemN2N) for question and answering task comprises data transformation, compiling the model, model fit and model assessment. The required packages are imported and the global variables are instantiated and initialised. The pre-processed data is in the form of inputs, questions and the corresponding answers and are split into training and validation datasets. Transformation between words and vectors (or tensors) are computed. The end-to-end memory architecture is setup in the model compilation step. The fitted model is trained and evaluated during training with the validation data. Note that for simplicity the expansions in the context were transformed as unigrams.

3.3 Transformer models

[Vaswani et al., 2017] proposed the Transformer and demonstrating that *(self-)attention is all we need* to remove encoder-decoder recurrent model layers and improving parallelisation. The paper acknowledged that end-to-end memory networks "perform well on simple-language question answering and language modeling tasks". Transformer models are deep learning models that are used extensively for natural language processing tasks; and as [Rogers et al., 2021] puts it as - "BERT has become a ubiquitous baseline in NLP experiments".

The model architecture proposed for this study is to fine-tune pre-trained transformers for question answering tasks. Inputting of question and context packages as a single sequence, along with answer start and answer end vectors are used during the model fit fine-tuning process. The questions contain the acronym short-form (SF) and the context contains the acronym expansions in long-form (LF); and the words around the acronym expansion disambiguates the various acronyms in-context.

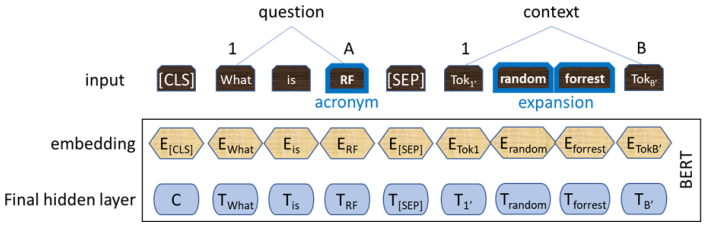


FIGURE 3.2: Question and context pair inputs and BERT for extractive question answering task

In Figure. 3.2 illustrates the data ingestion into BERT, for example, where the input data is pair of question and context contain the acronym short-form is ingested during the fine-tuning of the pre-trained transformer. As well as the acronym expansion character start and end positions, and the ground truth acronym expansion (and length of text).

In Algorithm 2 the transformer fine-tuning workflow for question and answering task comprises accessing a powerful pre-trained transformer model and it's associated tokenizer, preparing input data sequences as tensors (multidimensional arrays of numerical data), setting up optional extra model compile settings, model fitting (i.e. fine-tuning), and model assessment. The workflow is

Algorithm 2 A fine-tune a transformer model for question and answering task.

Import packages and Initialise all variables.
Import pre-processed data comprising training, validation, and test datasets.
Prepare data:
Load tokenizer, select padding side and context ID
Prepare data sequences:

- BERT sequence is [CLS] question [SEP] context [SEP] and set context ID as 1
- XLNet sequence is context <sep> question <sep> <cls> and set context ID as 0
- Locate acronym expansion start and end positions in the context

Preprocess data as tensors to be compatible with TensorFlow (TF), for example.
Load pre-trained model
Compile Model (optimiser, loss, metrics)
Fit Model (inputs), *meaning fine-tune pre-trained model*
Minimise valuation loss and Save best only.
Early stop with patience of 5.
Save model files. Garbage collection.
Assess Model (model, assessment_data):
Generate predictions for validation, and test data.
Compute SQuAD metrics: exact match and F1 score

sourced and adapted from huggingface[2] .
Starting comprise installing and importing the required packages, and initialising the global variables. Prepare inputs by preprocessing question and context pair data using the model's tokenizer. Now depending on the transformer model's architecture specifically build the required pair of sequences:

- Establish the order of context and question sequences
  - (for example, BERT sequence [CLS] A [SEP] B [SEP] and
  - for example, XLNet sequence A <sep> B <sep> <cls>), and
- In turn select the appropriate padding side and
- Discern the context ID label (and for example, 1 for BERT, and 0 for XLNet).

For this work, the context max padding length was selected as 96 tokens, and the stride (being the length of the previous sequence to be included in the overflow sequence, or overlap) calculated to the smaller power of 2 of 67.5% of half the max context length - so 96 max padding length yields an stride overlap of 32 tokens (whereas context of 384 tokens, for example, would result in a 128 token stride).
The pre-processed data is in the form of a paired sequences, and training inputs are complemented with labelled start and end token positions (to learn how to extract the acronym expansions). The offsets of the context are tracked and equal the context ID for the model validation and (unseen) test inputs; and the question token offsets are set to **None** since we only want to predict tokens in the context during the post-processing phase.
Compiling the transformer models for fine-tuning the following hyperparameters were used during the training:

[2] NLP Course Documentation - Question answering

- optimizer: 'name': 'Adam', 'learning_rate': 5e-05, 'decay': 0.0, 'beta_1': 0.9, 'beta_2': 0.999, 'epsilon': 1e-07, 'amsgrad': False
- training_precision: float32

The exact match and f1 score metrics are sped up by only assessing the top predicted start and end logits and selecting the best logit score as the predicted acronym expansion (or answer). The metrics evaluate the predicted acronym expansion for both the validation data used during training, and on any unseen test data kept aside.

3.3.1 Individual Transformer Models

Transformer model documentation is readily available for the huggingface library and the following state-of-the-art models are particularly highlighted:

- 'bert-base-cased'
- 'roberta-base'
- 'xlnet-base-cased'

According to the huggingface written model card of the BERT[3] based model (cased) and based on [Devlin et al., 2018] the model has 110M parameters and is pre-trained on a large English data corpus 3300M words, from BookCorpus and English Wikipedia. BERT has two pre-trained unsupervised tasks of Masked Language Modelling (MLM) and Next Sentence Prediction (NSP)) which is a suitable for downstream question answering tasks. Model inputs are in the form: [CLS] A [SEP] B [SEP]

The huggingface written model card of the RoBERTa[4] base states the model size as 125M parameters. The paper [Liu et al., 2019a] elaborates the model was pre-trained on five datasets BookCorpus, Wikipedia, and additionally with 63M news articles called CC-News, OpenWebText, and subset of CommonCrawl referred to as Stories- totalling 160GB of text. And that the training objective differs by masking a random sample of tokens and learns bidirectional representations of the sentence. The model inputs are in the form:
<s> A <s><s> B <s>

The huggingface written model card of the XLNet[5] base cased states XLNet is based on "a novel generalized permutation language modeling objective" and model inputs are in the form:
A <sep> B <sep> <cls>
In the paper [Yang et al., 2019] they proposed a generalized autoregressive pretraining technique that learns contexts "by maximizing the expected likelihood over all permutations of the factorization order" meaning all tokens are predicted but in random order. Compared to autoencoder pretraining technique of randomly masking some of the tokens and aims to reconstruct the original data from the masked/corrupted input.

[3] bert-base-cased
[4] roberta-base
[5] xlnet-base-cased

We briefly introduce other enhanced model architectures of interest, with their huggingface alias as follows:

- 'microsoft/deberta-v3-base'
- 'distilbert-base-cased-distilled-squad'
- 'allenai/scibert_scivocab_cased'
- 'deepset/tinyroberta-squad2'

[He et al., 2020] proposed DeBERTa which used two techniques of disentangled attention in the encoder and a better mask decoder that was now aware of absolute positions of tokens. Pre-training dataset is 78GB from Wikipedia (12GB), BookCorpus (6GB), OPENWEBTEXT (38GB), and STORIES (31GB). [He et al., 2021] further improved performance with DeBERTv3 which combines DeBERTa and ELECTRA. The DeBERTa improvement is by a technique called replaced token detection. Additionally their new gradient-disentangled embedding sharing (GDES) technique helps improve pre-training by preventing tug-of-war effects of binary classification optimisation that occurs between the generator and the discriminator in ELECTRA. Their main contribution was GDES to solving the pre-training instability and inefficiency cause by the tug-of-war when simply trying to combine DeBERTa and ELECTRA.

The DistilBERT model was proposed by [Sanh et al., 2019] which has 40% less parameters, executes 60% faster and keeps 95% of BERT's GLUE language understanding performance. Furthermore DistilBERT has a been fine-tuned on [Rajpurkar et al., 2016] SQuAD v1.1.

SciBERT for the scientific domain from [Beltagy et al., 2019] improved downstream scientific NLP tasks such as sequence tagging, sentence classification and dependency parsing with SciBERT (when compared to base BERT). The scientific corpus sourced from Semantic Scholar on domains computer science (18%) and biomedical (82%) is 3.17B tokens.

TinyRoBERTa SQuAD2 is a distilled RoBERTa based on [Jiao et al., 2019] tinyBERT methods which does knowledge distillation (KD) from large teacher model to a small student model aimed at edge devices to make BERT-like NLP models available to mobile phones, for example.

3.3.2 Answering Research Questions

Evaluating the performance of XLNet could answer the question: Is there more potential to disambiguate acronyms using generalized autoregressive pre-training methods?

The transformer baseline will be the independent results of RoBERTa and SciBERT when compared to smaller forms tinyRoBERTa and DistilBERT SQuAD to answer the question: how do smaller main stream autoencoding based models perform?

Based on the final results, it may or may not necessitate the pursuit of adapting the [Zhong et al., 2021] ensemble model called hdBERT, in Figure 2.1. Applying an emsemble SDU model to

MMM data should answer the question: What performance can pre-trained transformer models deliver when applied to the MMM domain data in a zero-shot application?

Thus the proposed conceptual models are:

- *Conceptual model one:* From the individual transformers utilise a good performing compact model as the base for MMM domain data.
- *Conceptual model two:* If generalized autoregressive pre-training methods like XLNet on this dataset performs better than RoBERTa then conceptual model one could be applied and tested on the MMM domain data.

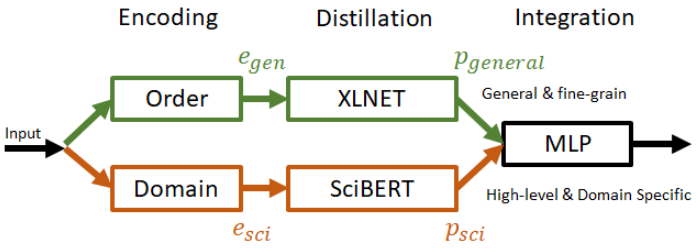


FIGURE 3.3: conceptual model two: adapted emsemble model based on hdBERT

In Figure. 3.3 illustrates data flow where the input sentence sample comprising acronym and expanded long forms is distributed to the various encoders: Order and Domain; their encoded outputs are inputs to pre-trained transformers XLnet and SciBERT and produce model predictions: $p_{general}$ and p_{Sci}. The model representations are integrated into the multiple layer perceptron network and sigmoid unit.

3.4 Evaluation Metrics

3.4.1 Metrics: EM and F1 score

Open dataset:

The primary metrics on the open dataset will be exact match (EM) and F1 score. F1 score begin the main metric.

Typically each object is associated with a binary label (L) where we test (obtain model predictions) for its correctness (meaning its relevance) and classify accordingly. The observed counts of true positive (TP) and the false positives (FP), false negatives (FN) and true negatives (TN) are then used in definitions of precision (p), recall (r), accuracy, F-score[6] (F_β, where $\beta = 1$ is the harmonic mean of precision and recall), and MCC.

Here we have defined that

[6] F_β measures the effectiveness of information retrieval: popular are $\beta = 2$ weighing recall higher than precision and $\beta = 0.5$ weighing recall lower than precision

1. real positives that are correctly predicted positives (a hit) are called true positives (TP);
2. real negatives that are wrongly predicted positives (false alarm) are called false positives (FP).
3. real positives that are wrongly predicted negatives (a miss) are called false negatives (FN);
4. real negatives that are correctly predicted negatives (right rejection) are called true negatives (TN);

These are the formula:

$$p = \frac{TP}{TP + FP} \tag{3.1}$$

$$r = \frac{TP}{TP + FN} \tag{3.2}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.3}$$

$$F_{\beta=1} = (1 + \beta^2)\frac{pr}{r + \beta^2 p} = \frac{2.TP}{2.TP + FN + FP} \tag{3.4}$$

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP + FP).(TP + FN).(TN + FP).(TN + FN)}} \tag{3.5}$$

For question answering exact match (EM) is a binary measurement. EM measures the percentage of predicted output that exactly matches or overlaps the ground truth answer. EM is the proportion of questions that are answered in exact same words as the ground truth.

F1 score is a harmonic mean of precision an recall. It is more lenient than the EM score and arguably applies human judgement of similarity between answers strings. In question answering each question, the precision is calculated as the number of correctly predicted words divided by the total words in the predicted answer. In question answering the recall is the number of correctly predicted words divided by the number of words in the ground truth answer. Thus the question answering F1 score measures the overlap between the predicted answer and the labelled answer. The F1 score is averaged among all the questions.

For question answering F1 score the predicted tokens and the ground truth tokens are assessed as follows:

1. TP is the number of tokens that are common between the predicted tokens and the ground truth answer tokens.
2. FP is the number of tokens that are in the predicted tokens but not in the ground truth answer tokens.

3. FN is the number of tokens in the ground truth answer tokens but not in the predicted tokens.

4. TN are true rejections (or no answer found in the context).

huggingface - Evaluate:

[Rajpurkar et al., 2016] introduced SQuAD or Stanford Question Answering Dataset from Wikipedia articles with question-answer pairs. And when the dataset is in the format of SQuAD then the huggingface evaluation metrics are readily available. The SQuAD[7] metric provides the essential measures of exact match and F1 score. Similarly the SQuAD v2[8] is available and will be utilised to evaluate the fine-tune pre-trained transformer models on EM and F1 score.

3.4.2 Domain dataset:

The domain specific data predictions will need to be evaluated by a survey for the domain experts if annotated data is not obtainable. 50 examples of the MMM data were selected and ground truth expansions were annotated - meaning small sample can be evaluated by EM and F1 score to have some basic results.

[7] HF metric card: SQuAD
[8] HF metric card: SQuAD v2

Chapter 4

Results

4.1 Baseline

4.1.1 End-to-End Memory

The baseline(LSTM based) model was trained and early stopped on the 31st epoch.

Appendix C.1 shows the model training and model validation plots of (a) model accuracy and (b) the F1 score over 31 epochs. Both training plots (accuracy and F1 score) are dampened when compared to the model validation plot; the accuracy and F1 score improve steeply up until the 9th epoch and slowed down its learning from the 12th epoch where the accuracy was 0.6992 (compared to validation accuracy was 0.7745). Learning incrementally improves until the 31st epoch at a trained accuracy of 0.798, which (or validation accuracy of 0.829). This gave evaluation test results of (weighted) F1 score of 0.7875 (or 78.75%), and a Cohen Kappa score of 0.8214 are achieved.
Table 4.1 has the baseline test results yielding an adequate F1 (weighted) score of 78.75% with a healthy accuracy of 0.8225 and an high agreement Cohen Kappa score of 0.8214 for the baseline model.

Table 4.1: Summary of the evaluation metrics of the baseline system

	Training & Validation		**Testing**
Metric	Result	Metric	Result
loss	0.7189	Cohen Kappa Score	0.8214
accuracy	0.7981	accuracy	0.8225
f1_score	0.8069	f1_score (weighted)	0.7875
val_loss	0.8655	f1_score (micro)	0.8225
val_accuracy	0.8227	f1_score (macro)	0.4546
val_f1_score	0.8305		
lr	0.0018		
	F1 score = 80.69%		F1 score = 78.75%

More information on the Baseline(LSTM based) end-to-end model training loss and scheduled learning rates are discussed in Appendix C.2

4.2 Model Results

4.2.1 Transformer Final Results

Several BERT family and XLNet models were selected and assessed with the same test dataset (meaning a portion of held-back or unseen data). The same hyper-parameters were used for all experiments.

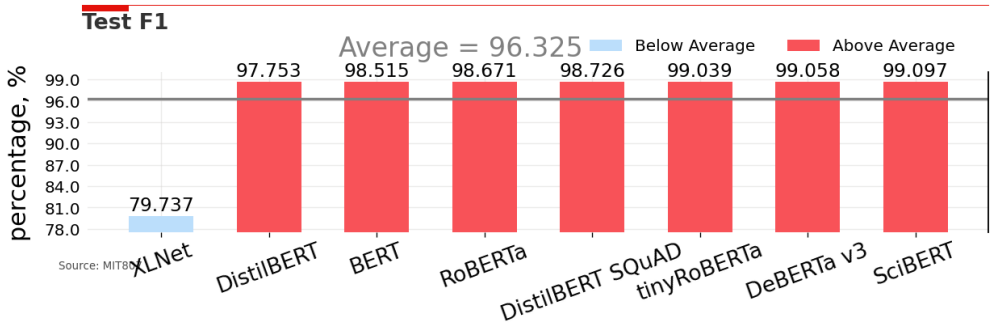


FIGURE 4.1: Individual F1 score results of fine-tuned transformer models for SciAD/ SDU dataset.
1st place SciBERT (f1=99.097%), 2nd place DeBERTa (f1=99.058%), and
3rd place tinyRoBERTa (f1=99.039%).
Average of the individual F1 score results was 96.325%

Figure 4.1 The final model results in ascending order of F1 score. The top performance with the (open) test data evaluation (of 11 245 examples) was achived by SciBERT (F1=99.097%), DeBERTa (F1=99.058%) took second place, and closely followed tinyRoBERTa (F1=99.039%). DistilBERT SQuAD (F1=99.097%) performed very well being the smallest model in terms of trainable parameters. And the average F1 score across the individual transformer models was 96.325%.

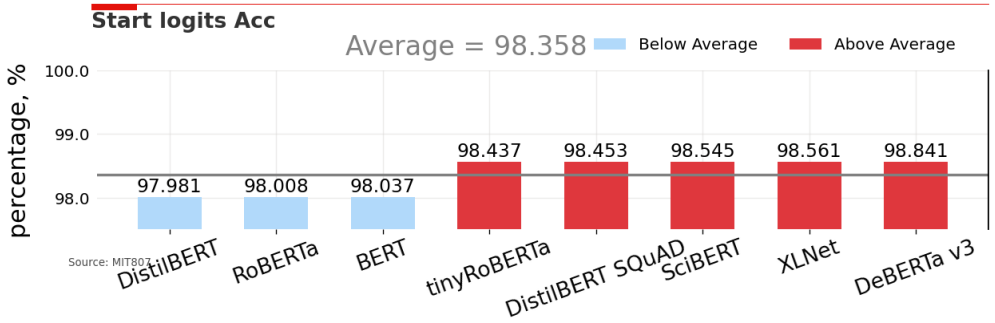


FIGURE 4.2: For the individual fine-tuned transformer models for SciAD/ SDU dataset.
XLNet in 2nd place with a start logits accuracy of 98.561%
Consider, XLNet had a low F1 score (F1 = 79.737%) in Figure 4.1.
The average of the individual start logits accuracy of 98.358%.

Figure 4.2 The start logit accuracy results in ascending order. The most interesting observation was XLNet in second place with regards to accuracy (of start logit). Unfortunately XLNet, with this data, the hyper-parameters, and the model compilation did not yield a strong F1 score (f1 = 79.737%) ; whereas the BERT family of models, in general, did perform well.

Highlights from Table A.2 of all individual transformer model results on SciAD/ SDU data, are shown in Table 4.2. Table 4.2 shows five individual transformer results of interest, namely tinyRoBERTa, BERT for reference, SciBERT, XLNet, and RoBERTa and ordered by model size of trainable parameters.

For the held-back test data, column 'Test F1', there are very impressive F1 scores from the BERT-like models. For this work XLNet was the outlier model with the lowest F1 score of 79.7373% yet had strong accuracy scores; SciBERT with an F1 score of 99.0972% is top for the short list of Table 4.2. tinyRoBERTa performed very well given it's small size in terms of 81.5 million trainable parameters (or 25.4% less parameters than SciBERT and 55.7% less parameters than DeBERTa). (From Appendix Table A.2 DeBERTa had the best overall metrics for exact match and F1 score.)

TABLE 4.2: Summary of fine-tuned pre-trained transformer results - Validation and Test datasets

Model	no. params (trainable)	Val EM	Val F1	Test EM	Test F1	Start logits Acc	End logits Acc
tinyRoBERTa	**81.5**	97.7323	98.9355	**97.8880**	**99.0393**	98.4368	97.9930
BERT	107.7	97.0476	98.4156	97.2432	98.5154	98.0370	98.0961
SciBERT	109.3	98.1058	99.1662	**97.9546**	**99.0972**	**98.5453**	**98.4738**
XLNet	116.7	60.8804	79.1452	61.4940	79.7373	**98.5606**	**98.6418**
RoBERTa	124.1	97.0920	98.7026	97.1098	98.6710	98.0078	97.7810

4.2.2 Domain Results

Table 4.3 lists the experimental results of a zero-shot application on MMM data. The SciAD/ SDU fine-tuned transformers, when applied to a small representation of MMM data, had two major findings:

- The F1 Scores on SciAD/SDU data typically halved for MMM data, and suggests that the SciAD/SDU models are not yet skilled enough for the MMM domain.
  - In the $F1_{base}$ column are some base transformer F1 scores from their respective zero-shot applications on the MMM data. It is observed that the SciAD/SDU fine-tuned transformers (on average 40.83%) performs twice as well as the base transformer (on average 19.25%).
- RoBERTa (1^{st} for MMM data with F1 score 52.20%) and SciBERT (3^{rd} for MMM data with F1 score 46.78%) potentially perform better than XLNet (last for MMM data with F1 score of 23.00%). DeBERTa (2^{nd} last) surprisingly was below the average F1 score

(40.83%) with F1 score of 34.60%, and note tinyRoBERTa (was 2nd) held a good 49.20% relative to RoBERTa.

TABLE 4.3: Summary of the Thin-slice Zero-Shot application on MMM data

	Exact Match	**F1**	$F1_{base}$
xlnet-base-cased	6	23.00	(13.58)
microsoft \ deberta-v3-base	16	34.60	
bert-base-cased	14	38.80	(23.77)
distilbert-base-cased-distilled-squad	18	41.20	
allenai \ scibert_scivocab_cased	10	46.78	(19.22)
deepset \ tinyroberta-squad2	16	49.20	
roberta-base	**24**	**52.20**	(20.42)
		Ave: 40.83	(Ave: 19.25)

We select 4 example MMM acronyms, namely PV, SIM, TT, and PD and highlight interesting observations from the selected examples listed in Appendix B.3.

NOTE: *In the case examples to follow, each sentence (or each box) is asked the question (which contains the acronym). The model then needs to answer the question by understanding the context in the box and returning the predicted acronym expansion; and disambiguate the question's acronym.*

Case Example 1: What is PV (process value)?

Portable Water Reservoir Inlet 2 Flow **process value**
Grey Water Supply Flow **process value**
Gland Service Water Flow From Tank 999-Tk-001 **process value**
Granulator Make-Up Water Tank Tk-001 Level **process value**

The context examples containing "PV" all end with the expansion "process value". The token count for the context spans from 6 to 8 tokens. Water is a common token and the only other token starting with "p" in 4 examples is "Portable". Figure 4.3 shows the model predicted

Ground truth	**roberta-base**	**tinyRoBERTa**
process value	process value	process
process value	**process value**	process
process value	**process value**	**process value**
process value	**process value**	process
scibert_scivocab_cased	**bert-base-cased**	**xlnet-base-cased**
Portable Water Reservoir	**process value**	process
process	**process value**	<empty>
process value	**process value**	process
process value	**process value**	process

FIGURE 4.3: Model predictions for the question: what is PV?

outputs for the question, "what is PV?" We observe the ground truth is exactly matched for BERT and RoBERTa, SciBERT exactly matched twice and tinyRoBERTa only exactly matched

once out of the 4 examples shown. XLNet at best predicted 1 token of the 2 ground truth tokens. SciBERT missed the mark by reporting back ”Portable Water Reservoir”. XLNet reported an empty string once.
The tokens ”process” and ”value” are arguably within the general domain and could explain why RoBERTa correctly expanded the acronym; the combination of the two tokens in context might be where the trouble lies, the phrase ”process value” might not be general anymore, but be MMM domain specific, or similar to the ”engineering” equivalent of ”Process Variable”.
In the SciAD dataset there are only two acronyms containing ”value” (SVD’: [’singular value decomposition’] and GVF’: [’generalized value function’]) and several containing ”process” (such as ’GPR’: [’gaussian process regression’] and ’DP’: [’dirichlet process’]), but not the combination of ”process value”.

Case Example 2: What is SIM (simulated)?

SXYZ Storage Silo 999-Sf-001 Level **Simulated**

The context example ends with the expansion ”simulated”. The acronym differs in that the three characters are the starting characters of the expansion. There are 6 tokens in the context.

Ground truth	**roberta-base**	**tinyRoBERTa**
Simulated	**Simulated**	**Simulated**
scibert_scivocab_cased	**bert-base-cased**	**xlnet-base-cased**
SXYZ Storage	Silo	SXYZ Storage

FIGURE 4.4: Model predictions for the question, ”what is SIM?”

Figure 4.4 shows the model predicted outputs for the question, ”what is SIM?” The RoBERTa’s were able to expand SIM to simulated, noting that SIM is moreso a prefix of the first 3 characters / shorthand form, rather than a typical acronym made from the first character of each sequential token.
In the SciAD data, the closest acronym ’SA’ (’simulated annealing’) would have been the model’s fine-tuned references to the expansion token ”simulated”.

Case Example 3: What is TT (temperature transmitter or (shorthand) temperature)?

PVR Fan Bearing Oil Supply - **Temperature** High Setpoint
PVR Fan Bearing Oil Supply - **Temperature** Output
Fan bearing **temperature** motor side axial High Setpoint
Fan bearing **temperature** motor side axial Output
Crystalliser PVR Fan 2 Hydraulic Oil **Temperature** High Setpoint
Crystalliser PVR Fan 2 Hydraulic Oil **Temperature** Output

The context examples have the expansion ”temperature” on either side of the center token. The acronym is two characters in length and the expansion is 1 token. Two of the contexts have

Ground truth	roberta-base	tinyRoBERTa
Temperature	Setpoint	point
Temperature	**Temperature**	**Temperature**
Temperature	**temperature**	temperature motor side axial
Temperature	**temperature**	temperature motor
Temperature	Setpoint	Temperature High Setpoint
Temperature	Oil Temperature	**Temperature**
scibert_scivocab_cased	**bert-base-cased**	**xlnet-base-cased**
Temperature High Setpoint	Fan Bearing Oil Supply	<empty>
Temperature Output	Fan Bearing Oil Supply	<empty>
temperature motor	motor side axial	temperature motor side
temperature motor	motor side axial	temperature motor side
Temperature High	Hydraulic	Crystalliser
Temperature Output	Hydraulic	Crystalliser

FIGURE 4.5: Model predictions for the question: what is TT?

punctuation, a hypen and where the expansion immediately follows.
Figure 4.5 shows the model predicted outputs for the question, "what is TT?" For our purposes, either an exact match of 'temperature' (in context) or the full 'temperature transmitter' are deemed correct. The RoBERTa's were able to expand TT as temperature the most times out of the 6 examples, where RoBERTa had 3 exact matches and tinyRoBERTa had 2 exact matches. SciBERT expanded temperature but was not aware that TT should be one token; instead SciBERT for 5 examples reported an incorrect second token immediately after temperature. BERT and XLNet predicted the most incorrect results.
In the SciAD data the acronym 'TS': ['temperature scaling', 'temperature - based sampling'] would have been the model's fine-tuned references.

Case Example 4: What is PD (pressure differential or differential pressure)?

Cyclone 1 **differential pressure** Input warning high

The context example has the expansion near the centre of phrases and comprises the two tokens "differential pressure". The acronym is two characters in length but the order may or may not differ to the expansion.

Ground truth	roberta-base	tinyRoBERTa
differential pressure	**differential pressure**	**differential pressure**
scibert_scivocab_cased	**bert-base-cased**	**xlnet-base-cased**
differential pressure Input warning	**differential pressure**	**differential pressure**

FIGURE 4.6: Model predictions for the question, "what is PD?"

Figure 4.6 shows the model predicted outputs for the question, "what is PD?" All the models except SciBERT got exact matches. SciBERT reports a 4 token expansion instead of 2 tokens. In the SciAD dataset, 'PD' gave the model the following expansion knowledge ['progressive disease', "prisoner's dilemma", 'pu - primary destination', 'positive definite', "parkinson's disease", 'pixel discussion'].

4.3 Predictability, Computability and Stability (PCS)

[Yu, 2020] presented a data science life cycle framework to provide results that are responsible, reliable, reproducible and transparent, and in particular with human judgments or choices made. This framework of predictability, computability and stability was applied to this work in the following way:

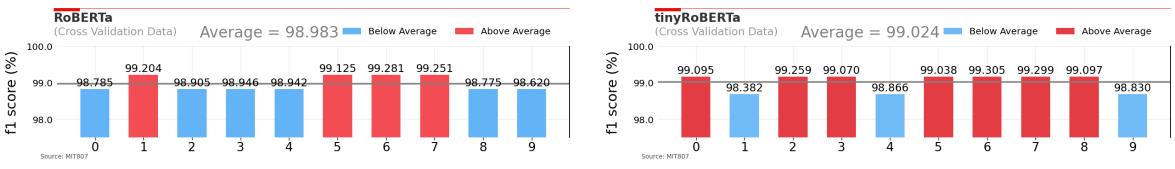
- model evaluation results are based on held-out test data and is compared to model fine-tuning results;
- cross-validation was performed to support the final model results;
- model parameters and batch sizes were selected to utilise readily available resources such as Google Colab (free 12Gb GPU VRAM); and
- an exploratory data analysis performed to support data pre-processing choices.

4.3.1 Predictability

In addition to the results of test (held-out) data of Table A.2 of all individual transformer models for SciAD/ SDU data, we evaluated the model predictions using cross-validation (CV) datasets.

Cross-Validation (k=10, not shuffled) Four models, RoBERTa, tinyRoberta, DistilBERT and SciBERT, were subjected to a k-fold cross-validation process to estimate the skill of the models which all performed very well for F1 scores.

Extracted from Appendix Table A.3, Table 4.4 shows the results of various transformers using a k-fold cross-validation dataset. DistilBERT had the lowest cross-correlation F1 score mean of 98.751% and the best cross-correlation F1 score mean of 99.024% was from tinyRoBERTa. Note that tinyRoBERTa did have the largest standard deviation of 0.166% and SciBERT the least standard deviation of 0.173%.
Furthermore the model evaluation on the test data reported similar numbers, DistilBERT still had the lower of the four models, but RoBERTa (99.050%) took the lead over tinyRoBERTa (99.006%) by a narrow margin. Evaluating the means, which summarises model skills, one sees the model evaluation values are trustworthy since the difference is quite small; further we expect the means to be a less biased result.



(A) RoBERTa k-fold validation

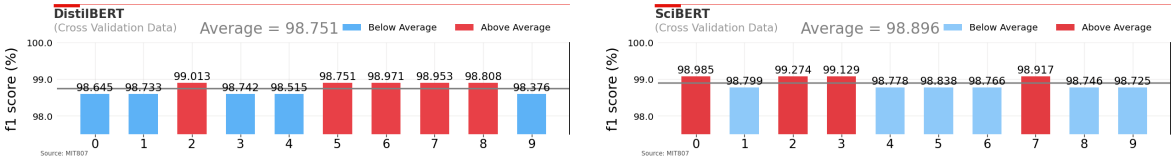
(B) tinyRoBERTa k-fold validation

FIGURE 4.7: Cross-validation results (k=10) for RoBERTa and tinyRoBERTa.
The average over the 10 folds was a F1 score of 98.983% for RoBERTa.
The average over the 10 folds was a F1 score of 99.024 for tinyRoBERTa.

TABLE 4.4: Summary of selected pre-trained transformers - k-fold cross-validation datasets

validation	RoBERTa	tinyRoBERTa	DistilBERT	SciBERT
0	98.7845	99.0949	98.6450	98.9855
...	...	...	...	...
9	98.6201	98.8301	98.3762	98.7251
Val. Ave.	98.983	99.024	98.751	98.896
test	RoBERTa	tinyRoBERTa	DistilBERT	SciBERT
0	98.8820	99.0729	98.5830	99.0181
...	...	...	...	...
9	98.9338	99.0730	98.5594	98.9550
Test Ave.	99.050	99.006	98.723	98.875

In Figure 4.7 (a) RoBERTa and (b) tinyRoBERTa are side-by-side for CV comparison. tinyRoBERTa achieved the higher F1 score mean of 99.024% across the 10 folds than RoBERTa's 98.983%, but RoBERTa was more stable in the sense that its F1 score standard deviation was 0.2128% (compared to 0.2640% SD). RoBERTa peaked on the 6^{th} fold with 99.281%, and tinyRoBERTa had it's best result of 99.305% on the same fold.



(A) DistilBERT k-fold validation (B) SciBERT k-fold validation

FIGURE 4.8: Cross-validation (k=10) results for SciBERT and DistilBERT.
The average over the 10 folds was a F1 score of 98.896% for SciBERT.
The average over the 10 folds was a F1 score of 98.751 for DistilBERT.

In Figure 4.8 (a) DistilBERT and (b) SciBERT performed well for their respective CV results. On average DistilBERT F1 score was 98.751% (with standard deviation 0.1921%) and SciBERT F1 score was on average 98.869% and was more stable with a standard deviation of 0.1735%. DistilBERT preformed its best on the 2^{nd} fold, but did not match SciBERT's best 2^{nd} fold of 99.274%.

Cross-Validation (k=5, shuffled)

Table A.4, has been highlighted in Table 4.5 where the 5 fold split of all the data. The validation data was shuffled and selected in the ratio of 85:15 for training and validation. The average validation F1 scores of tinyRoBERTa of 99.130% compared to RoBERTa's F1 score average of 99.118%. The shuffled 5 fold CV results are similar magnitudes to the original individual test F1 score in Table A.2.

TABLE 4.5: Summary of selected pre-trained transformers - k-fold (k=5) CV data, shuffled

validation	RoBERTa	tinyRoBERTa	DistilBERT	SciBERT
Val. Ave.	99.118	99.130	98.888	98.863

CV Exploratory data analysis (EDA) The SciAD/ SDU dataset comprising development and training files was concatenated and equalled 56 223 examples. The data was split into training

and test data. The training data was further split to obtain validation and cross-validation data sets.

Table 4.6 for the two CV investigations,

- The model dataset was subjected to a k-fold split of k=10. The training data plus the validation data was re-split into the ratio 90:10 (or 40 480:4 498 examples); and the test dataset remained the same as the model dataset.
- For the k-fold cross-validation split, k=5 and all model datasets were re-merged and then re-split into a ratio of 85:15 (or 47 789:8 433 examples). Validation data was selected randomly.

TABLE 4.6: Summary of the SciAD/SDU data set splits

	Model	**10 Fold CV**	**5 Fold CV**
	Examples	Examples	Examples
Training	38 231	40 480	47 789
Validation	6 747	4 498	8 433
Testing	11 245	11 245	
	Total = 56 223	Total = 56 223	Total = 56 223

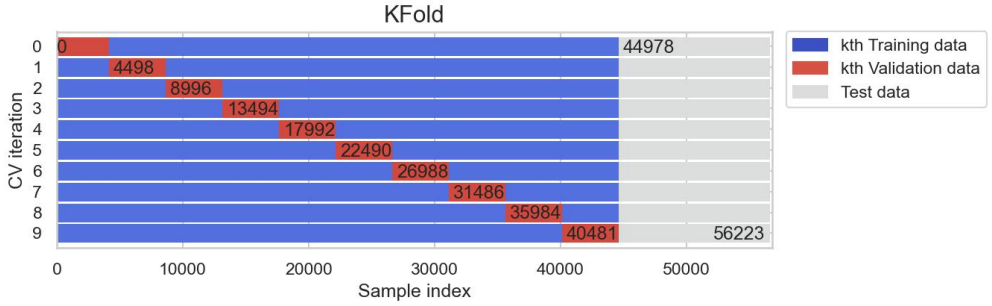


FIGURE 4.9: Cross-Validation (CV) - k-fold data groupings, k=10, no shuffle, one test dataset. Training data 40 480, Validation data 4 498, and test data 11 245 examples. Total examples per fold was 56 223.

Figure 4.9 The cross-validation data changes across the 10 k-folds for model training and validation, and test data which is unseen to the model is evaluated.

Figure 4.10 display how all the validation data was shuffled and selected as the 5 folds of training and validation data used to cross-validate the models.

4.3.2 Computability

Computability are key considerations and the minimum requirement to generate results is hardware accessibility. Computer memory (RAM), and in particular graphics processor video memory (VRAM) were limiting factors impacting successful modelling or modelling times. The computer memory availability had an influence on:

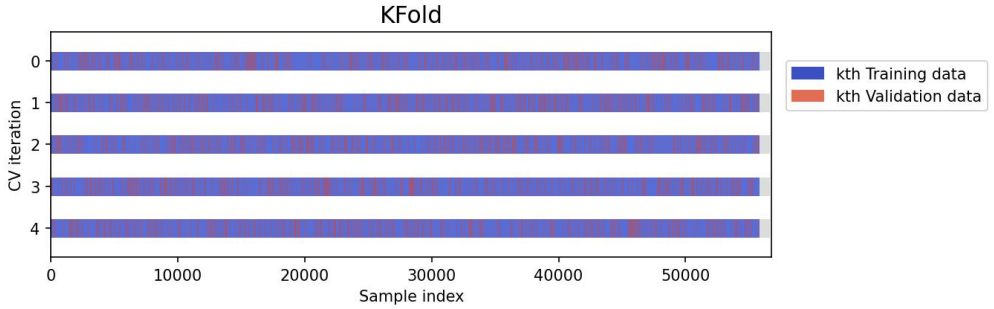


FIGURE 4.10: Cross-Validation (CV) - k-fold data groupings, k=5, shuffled.
47 789 Training and randomly sampled 8 433 Validation examples used for each of the 5 folds.
Total examples per fold was 56 223.

- model size parameters selection for LSTM baseline
- model prediction batch size for transformers

to avoid mainly out of memory (OOM) scenarios of VRAM.

Fine-tuning transformers required at least 3Gb of VRAM. The more demanding transformers were DeBERTa requiring at least 5Gb VRAM (and 10-12Gb recommended); and XLNet modelling and evaluation stages were split and run into two notebooks to work around the available hardware RAM and VRAM.

4.3.3 Stability

Exploratory data analysis (EDA) The SciAD/ SDU dataset comprises input (or context) sentences with an ambiguous acronym and a dictionary of potential acronym expansions (or long-forms). The development and training data is 56 223 inputs with four features: input sentence unique identification, acronym position, acronym expansion, and the input sentences. So the data shape assessed is (56 223, 4).

In the illustrative example below, the ambiguous acronym ('NN' shown underlined and bold) in the input sentence should be predicted and expanded as 'neural network' to have the correct meaning:

> *Input (sentences):* The proposed model uses **NN**.
>
> *Input (dictionary):* 'NN': ['neural network', 'nearest neighbor']
>
> *Output:* neural network

In the SciAD/ SDU example below, the *input sentence* with *id* 'TR-49802' has the *acronym* 'RF' *positioned* at index 16 is expanded as 'random forest":

> *Input (sentences):* Then , for our classification purposes , we apply three different classification algorithms : DT , **RF** and NN , accomplished with Scikit - Learn .

Input (dictionary): 'RF': ['random forest', 'radio frequency', 'regression function', 'regression forest', 'register file']

Output: random forest

The dictionary of 732 ambiguous acronyms are from the scientific domain. There are 2 308 expansions. The 'CS' acronym has the most potential expansions of 20: 1. 'computer systems', 2. 'computer science', 3. 'clonal selection', 4. 'connection size', 5. 'computational science', 6. 'centralized solution', 7. 'compressive sensing', 8. 'core semantics', 9. 'coordinated scheduling', 10. 'charging station', 11. 'constraint solver', 12. 'conventional sparsity', 13. 'compressed sensing', 14. 'critical section', 15. 'common subset', 16. 'content store', 17. 'case - sensitive', 18. 'consensus score', 19. 'code - switching', 20. 'cluster - specific'.

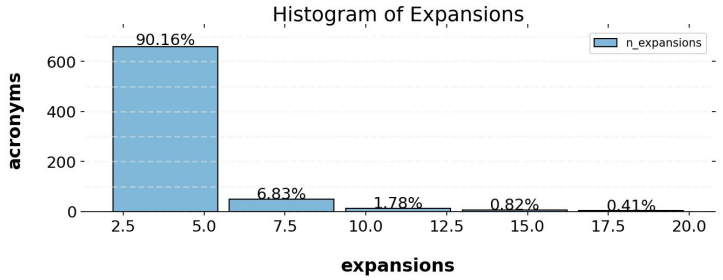


FIGURE 4.11: Histogram of acronym expansions.
660 (or 90.16%) of acronyms have up to 5 expansions. 437 acronyms have 2 possible expansions. The 55 (or 2.38%) most ambiguous acronyms have between 17 and 20 expansions.

Figure 4.11 shows the distribution of expansions. On average the acronyms have 3 potential expansions. The average is pulled by the top most ambiguous acronyms. The top three most ambiguous acronyms are 'CA', 'SC', and 'CS' which have 17, 18 and 20 expansions respectively. A high-level breakdown of the acronyms and their potential expansions are as follows:

- 660 (90.16%) acronyms have up to 5 expansions, of which 437 have 2 expansions. These 660 acronyms account for 1664 of 2308 expansions (72.09%).
- 50 (6.83%) acronyms have 5 to 9 expansions (summing to 362 expansions, or 15.68%)
- 13 (1.78%) acronyms have 9 to 12 expansions (141 expansions, or 6.11%)
- 6 (0.82%) acronyms have 12 to 16 expansions 86 expansions, or 3.73%)
- 3 (0.41%) acronyms have 17, 18 and 20 expansions (summing to 55 expansions, or 2.38%)

Figure 4.12 shows the 99^{th} percentile distribution of the acronym positions. Acronym positions are positions (or indices) in the input sentences. It is observed that the first two histogram bins account for 87.40% (57.89% and 29.51%). The middle bin is 9.40% and the remaining two bins are 3.20%. Furthermore in 95.30% of all cases the acronym is positioned less or equal to the 32^{nd} position of the sentences, and 99.30% by the 48^{th} position.

Insights from Figure 4.12 that 95.30% of acronyms in the open dataset are positioned less or equal to the 32^{nd} position of the sentences, the following data preprocessing choices were made:

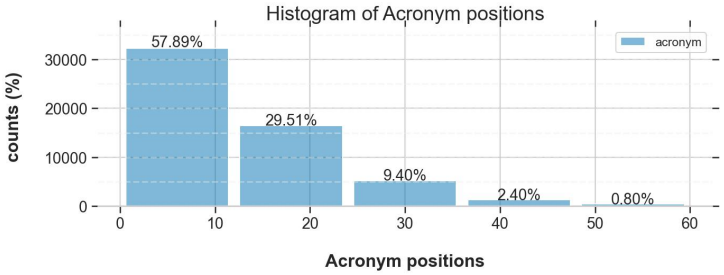


FIGURE 4.12: Histogram of Acronym positions.
95.30% of the acronym expansion are not more than the 32nd position of the sentences.

- Limit the model input sentences to 32 ”tokens” and as far left of the acronym position as possible.
- Input sentences less than 32 ”tokens” are padded from the left.

.

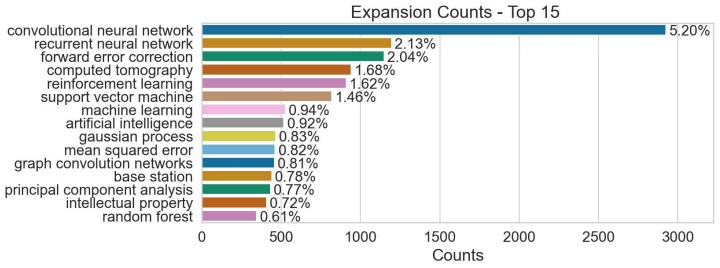


FIGURE 4.13: Histogram of Acronym expansion counts.
The most counts are convolution neural network (CNN).
The second most occurring expansion is recurrent neural network (RNN).
The top 15 acronym expansions seem to have a data analytic and/or data science theme.

Figure 4.13 shows the top 15 acronym expansion counts. 5.20% of the acronyms can be expanded as ’convolutional neural network’; whereas ’machine learning’, ’artificial intelligence’ and ’random forest’ are 0.94%, 0.92% and 0.61% respectively. In my opinion, the top 15 acronym expansions seem to have a dominant theme of *data science*.

The vocabulary size of the open dataset is 44 609 tokens (comprising English words” and acronyms).

4.3.3.1 Open Dataset

For SciAD/ SDU datasets the context length was capped to 32 tokens, based on the majority of the acronym positions being equal to or less than 32 (Figure 4.12). The following data cleaning choices were made for the baseline data set:

- Replace punctuation as $<punc>$
- Replace quantity text pattern as $<num>$

- Replace none alphabet characters with an underscore, replace entire underscored tokens as *<unk>*

Similarly the main data set used for transformer fine-tuning, the following choices were made:

- Quantity text was replaced with generic '123' string.
- None alphabet characters were replaced by underscores and entire underscored words removed from the context.
- Removed url and consequently removed hyphenated words.
- special cases: " - " or "o - d " as empty string; and " 's" to "s"

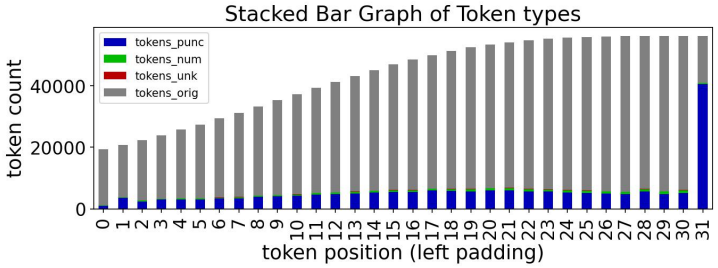


FIGURE 4.14: Histogram of Acronym expansion positions in the input contexts. 32 token lengths are classified as punctuation, number string, unknown (and typically not an alphabet character), or are original token.

Figure 4.14 shows the token types counts per token position in context. It is observed that left padding is applied due to the token type counts being least at position 1 (or 0 index in the figure); and most at position 32 (being 31 index in the figure). Further it can be seen that the context remain the original tokens (in grey), or were classified as punctuation (in blue); and that the context most likely ends with a punctuation mark, as indicated by the large count in the 31 index token position.

Chapter 5

Discussion

We asked the question, do sensor tag descriptions have expansions that match exactly or at least are close to acronyms in the sensor tag? So, we embarked on the task of determining whether one can measure if acronyms can be expanded for the MMM domain by using models knowledgeable in the scientific language (e.g. SciBERT) or in the general language (e.g. RoBERTa or XLNet) domain. This would determine if NLP machine pre-trained transformers could be applied to the MMM domain which have a low resource setting and little (or no) acronym dictionaries.

The high-level research questions we set out to investigate in this work are as follows:

- What performance can pre-trained transformer models deliver when applied to the MMM domain data in a zero-shot application?
- Is there more potential to disambiguate acronyms using generalized autoregressive pre-training methods?
- How do smaller main stream autoencoding based models (such as BERT-like models) perform?

The key findings are as follows:

- BERT-like (Main-stream autoencoding based) models are able to accurately predict SciAD/ SDU expansions after fine-tuning. But SciAD/ SDU data seems inadequate for the MMM domain.
- XLNet (a generalized autoregressive pretrained model) when fine-tuned with SciAD/ SDU data contrasts with other work results; XLNet performed the worst in the F1 score metric.
- Small transformers seem to perform well for domain specific question and answering tasks of acronym expansions in the SciAD/ SDU domain (comprising computer science and biomedical acronyms). Ensemble models don't seem to be required for this task at this point.

The End-to-end LSTM based models gave a baseline weighted F1 score result of 78.75%, for unigram predictions only. The accuracy was 0.8225 and the Cohen Kappa score was 0.8214. The high accuracy suggests that the ground truth compared to the predicted expansion matched exactly in 82.25% of cases, and the high Cohen Kappa score implies that there was high inter-annotator agreement again between the ground truth expansion and the predicted expansion. The F1 score is the contributions of precision and recall, and suggests that the model output quality is good since the weighted F1 score was 78.75% and takes into account the expansion support (or number of actual occurrences of expansions in the dataset), meaning we assigned a greater contribution to expansions that have more examples in the dataset. A low macro F1 score of 45.46% suggests the imbalanced data influenced this metric result. We saw evidence of imbalanced data in the acronym expansion counts where most expansions were 'convolutional neural network' at 5.20% and at 15$^{\text{th}}$ place to suggest a decay of counts is 'random forest' which accounted for 0.61% of expansions.

For SciAD/ SDU data BERT-like models performed extremely well where the average F1 score of the individual transformer models achieved was 96.325%. The two most notable individual results were from SciBERT being the top performer with a F1 score of 99.0972% and tinyRoBERTa with a F1 score of 99.0393% because of its model parameter size. These F1 scores are supported by the strong exact match results. Furthermore we tend to see a similar pattern emerge for a small representation of the MMM data for which the top three F1 scores were SciBERT (46.78%), tinyRoBERTa (49.20%) and RoBERTa (52.20%); for reference sake the top performer for the MMM data was the latter. The performance of the BERT-like models called for further support to displace the possibility of model over-fit (due to abnormally high F1 score results). K-fold cross-validation was done to support the original validation and test data F1 score results. Now, for the two transformers SciBERT and tinyRoBERTa, the results seem to reveal the following insights:

- *SciBERT* BASE Test F1 score change from 16.8625% to 99.0972% suggests that a lot of the SciAD knowledge was learnt.
- *tinyRoBERTa* BASE Test F1 score was the highest at 52.8226%, and the change to 99.0393% suggests the least SciAD knowledge was learnt.
- *Both SciBERT and tinyRoBERTa* (test) F1 scores are supported by high exact match scores and suggests that the SciAD models can predict very well in computer science and biomedical contexts.
- *Both SciBERT and tinyRoBERTa* halved in F1 score for MMM data and suggests that the MMM acronyms differ to SDU acronymns, and/or the context of MMM differs to SDU context.

XLNet did not perform well and contrasts to other work done, this suggests that further opportunities are available to understand the conflicting results, in the hope of unlocking the potential benefits that XLNet ought to have over BERT.

Limitations: The limitations to this work starts with the open SciAD dataset - only 732 acronyms biased to computer science and biomedical fields are known to the models from a

limited number of English research papers. Similarly the very small sample of MMM data does not fully represent the entire domain. Thus the limited amount of SciAD acronyms available might explain why the zero-shot MMM data results yielded half the F1 scores of the SCiAD/ SDU F1 scores.

For the SciAD/ SDU domain, even though the models' F1 scores were very high it might not be generalised enough. Three thoughts suggests this, firstly the computer science and biomedical domains are continuously growing as the science grows in knowledge - so future acronyms and uses in context are unseen. Secondly scientific document understanding is not generalised enough to handle sciences outside the sphere of computer science or biomedical acronyms - as seen when applying it to MMM data. For example PV, TT and SIM are not in the SciAD acronym dictionary; PD is, but the expansions are not the MMM expansions "differential pressure" or "pressure differential". Thirdly, only 6 786 English papers were used as the corpus and on average acronyms had 3 expansions, so the exposure to how the acronyms ought to be disambiguated might be limited to the limited amount of training examples.

An improvement to ensure the model is not over-fitting (or under-fitting) is to plot both the training and validation loss and compare the difference (or gap) between the two plots - unfortunately, at the time of modelling, only the training plot data was captured. Notes captured in Appendix C.1.2

Finally, we establish that pre-trained transformers can be fine-tuned to perform well in the boundaries of the domain data (in this case SDU), and does not necessarily transfer easily to other domains like MMM.

For the domain-specific MMM dataset, F1 scores were only able to get marginally larger than 50% using RoBERTa in conjunction with the SciAD dataset (where it attained 52.20%). Note furthermore that MMM results may be over-exaggerated since only 50 examples of MMM data were assessed, compared to the more than 60,000 sentences contained in SciAD. Scores, in general, were considerably lower than the results of the scientific document understanding (SciAD) open dataset, as is to be expected. In descending order, RoBERTa, tinyRoBERTa and SciBERT are the top three pre-trained language models models for acronym disambiguation using SciAD dataset. Zero-shot learning abilities are mostly due to the empowerment of the pre-training stage on vast amounts of good quality data. The fine-tuning of the pre-trained language models (PLM)s on the SciAD dataset helped the PLMs to be more aware of some MMM knowledge. We realise more MMM acronyms and their expansions in context is required to improve any PLMs performance.

We realise that we have not unlocked the potential of XLNet to disambiguate acronyms and that there are opportunities to do so in any future work. We discover that a small model could be used for the MMM domain, and in particular, tinyRoBERTa.

Chapter 6

Conclusion

In conclusion, in this work, we presented SciAD/ SDU fine-tuned transformers that can disambiguate acronyms within Scientific document understanding (SDU) context very well and is a stepping stone to being used in the Mining, Metals & Minerals (MMM) domain in future. We foresee that there is still opportunity to unlock the benefits of methods such as XLNet. We note the value that a small model could be used for the MMM domain, and in particular tinyRoBERTa.

Recommendations and future work are to embark on MMM domain data collection efforts, creating synergies to pre-process new MMM domain data from other acronym tasks to form an open MMM_{AD} dataset, and in turn run future experiments starting with smaller transformers.

Collect domain data: The process for which the SciAD dataset was collected could be applied to any MMM domain open data sources. The same could be said to collect similar domain data in disciplines such as process engineering, chemical engineering, and manufacturing. Having access to high-quality domain data and fine-tuning accordingly should improve any model's skill in those particular domains - as has been seen for the SciAD/ SDU dataset.

Preprocess new data and other acronym tasks: With more raw MMM data downstream, tasks such as Abbreviation Definition Identification (ADI), and/or Acronym Identification (AI) could be used to collect the essential acronym short- and long forms for a comprehensive dictionary. Additionally the context can be collected for acronym disambiguation (AD). With a large amount of acronyms and a large amount of context containing the acronym expansions, models could become more knowledgeable after model training.

Recommended baseline transformer: Furthermore, it seems like the smaller models such as tinyRoBERTa suffice for baseline experiments due to its size, without the expense of performance.

Bibliography

[Beltagy et al., 2019] Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *EMNLP*. Association for Computational Linguistics, 2019. URL https://www.aclweb.org/anthology/D19-1371.

[Charbonnier and Wartena, 2018] Jean Charbonnier and Christian Wartena. Using word embeddings for unsupervised acronym disambiguation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2610–2619, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics. URL https://aclanthology.org/C18-1221.

[Chiu and Baker, 2020] Billy Chiu and Simon Baker. Word embeddings for biomedical natural language processing: A survey. *Lang. Linguistics Compass*, 14, 2020.

[Church and Liu, 2021] Kenneth Church and Boxiang Liu. Acronyms and opportunities for improving deep nets. *Frontiers in Artificial Intelligence*, 4, 2021.

[Dai et al., 2019] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019. URL https://arxiv.org/abs/1901.02860.

[Devlin et al., 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

[G.K. et al., 2003] Savova G.K., Coden A.R., Sominsky I.L., Johnson R., Ogren P.V., de Groen P.C., and Chute C.G. Word sense disambiguation across two domains: biomedical literature and clinical notes. *J Biomed Inform.*, 41(6):1088–100, March 2003. doi: 10.1016/j.jbi.2008.02.00.

[He et al., 2020] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*, 2020.

[He et al., 2021] Pengcheng He, Jianfeng Gao, and Weizhu Chen. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*, 2021.

[Huang et al., 2019] Kexin Huang, Abhishek Singh, Sitong Chen, Edward T. Moseley, Chih ying Deng, Naomi George, and Charlotta Lindvall. Clinical xlnet: Modeling sequential clinical notes and predicting prolonged mechanical ventilation, 2019.

[Huang et al., 2020] Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission, 2020.

[Jiao et al., 2019] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*, 2019.

[Kovaleva et al., 2019] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of bert. *arXiv preprint arXiv:1908.08593*, 2019.

[Li et al., 2018] Yang Li, Bo Zhao, Ariel Fuxman, and Fangbo Tao. Guess me if you can: Acronym disambiguation for enterprises. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1317, 2018.

[Liu et al., 2019a] 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. *CoRR*, abs/1907.11692, 2019a. URL http://arxiv.org/abs/1907.11692.

[Liu et al., 2019b] 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, 2019b. URL https://arxiv.org/abs/1907.11692.

[Ma et al., 2020a] Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. Charbert: Character-aware pre-trained language model. *CoRR*, abs/2011.01513, 2020a. URL https://arxiv.org/abs/2011.01513.

[Ma et al., 2020b] Wentao Ma, Yiming Cui, Chenglei Si, Ting Liu, Shijin Wang, and Guoping Hu. Charbert: Character-aware pre-trained language model. *arXiv preprint arXiv:2011.01513*, 2020b.

[Moradi et al., 2021] Milad Moradi, Kathrin Blagec, and Matthias Samwald. Deep learning models are not robust against noise in clinical text. *arXiv preprint arXiv:2108.12242*, 2021.

[Navigli, 2009] Roberto Navigli. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69, 2009.

[Oh et al., 2022] SH. Oh, Kang M., and Lee Y. Protected health information recognition by fine-tuning a pre-training transformer model. *Healthc Inform Res.*, 28(1):16-24, 2022. doi: 10.4258/hir.2022.28.1.16.

[Pan et al., 2021] Chunguang Pan, Bingyan Song, Shengguang Wang, and Zhipeng Luo. Bert-based acronym disambiguation with multiple training strategies. *arXiv preprint arXiv:2103.00488*, 2021.

[Rajpurkar et al., 2016] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[Rogers et al., 2021] 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, 2021.

[Sanh et al., 2019] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC*^{2} *Workshop*, 2019.

[Sukhbaatar et al., 2015] Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. *arXiv preprint arXiv:1503.08895*, 2015. doi: https://doi.org/10.48550/arXiv.1503.08895.

[Vaswani et al., 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Veyseh et al., 2020] Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. What Does This Acronym Mean? Introducing a New Dataset for Acronym Identification and Disambiguation. In *Proceedings of COLING*, 2020.

[Weston et al., 2014] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *arXiv preprint arXiv:1410.3916*, 2014.

[Wu et al., 2017] Yonghui Wu, Joshua C Denny, S Trent Rosenbloom, Randolph A Miller, Dario A Giuse, Lulu Wang, Carmelo Blanquicett, Ergin Soysal, Jun Xu, and Hua Xu. A long journey to short abbreviations: developing an open-source framework for clinical abbreviation recognition and disambiguation (card). *Journal of the American Medical Informatics Association*, 24(e1): e79–e86, 2017.

[Xu et al., 2012] Hua Xu, Peter D Stetson, and Carol Friedman. Combining corpus-derived sense profiles with estimated frequency information to disambiguate clinical abbreviations. In *AMIA annual symposium proceedings*, volume 2012, page 1004. American Medical Informatics Association, 2012.

[Yang et al., 2019] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019. URL http://arxiv.org/abs/1906.08237.

[Yu, 2020] Bin Yu. Veridical data science. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 4–5, 2020.

[Zhong et al., 2021] Qiwei Zhong, Guanxiong Zeng, Danqing Zhu, Yang Zhang, Wangli Lin, Ben Chen, and Jiayu Tang. Leveraging domain agnostic and specific knowledge for acronym disambiguation. *arXiv preprint arXiv:2107.00316*, 2021.

Appendix A

Results

Baseline Transformer performance: Table A.1 shows the individual transformer results with no fine-tuning. The average of the base transformer exact matches was 5.552% and the average f1 score was a low 22.635%. The best performance contribution can from tinyRoBERTa with an exact match of 26.211% and f1 score of 52.822%. The worst base transformer performance came from XLNet with an exact match of 0.111% and f1 score of 3.620%.

Table A.1: Summary of base pre-trained transformer results - Validation and Test datasets

Model	no. params (trainable)	Val EM	Val F1	Test EM	Test F1
Base DistilBERT SQuAD	65.19	16.5318	43.8560	16.3851	43.6570
Base DistilBERT	66.36	0.4713	18.8452	0.57803	18.7013
Base tinyRoBERTa	**81.53**	**26.3673**	**53.4349**	**26.2116**	**52.8226**
Base BERT	107.72	0.3646	17.9283	0.33348	18.2428
Base SciBERT	109.35	0.4980	17.0822	0.3779	16.8625
Base XLNet	116.72	0.1512	3.4441	0.11116	3.6204
Base RoBERTa	124.06	0.3646	15.0665	0.24455	14.8858
Base DeBERTa v3	183.76	0.2045	12.5361	0.1779	12.2921

Fine-tuned Transformer performance: The Table A.2 shows the individual transformer results from the training validation data and unseen test data evaluations, after fine-tuning and using SciAD/SDU data.

The model results are ordered by model size of trainable parameters. For the unseen test data, column 'Test F1', there are very impressive F1 scores from all the BERT like models, and DistilBERT having the lowest 97.7532% of the BERTs. XLNet strangely was the outlier model with the lowest F1 score of 79.7373%, for this work.

The top performance was achieved by SciBERT with F1 of 99.0972%, DeBERTa in second and tinyRoBERTa comes third with F1 of 99.0393%. DeBERTa had the best overall metrics for exact match and F1 score. XLNet had strong accuracy scores.

The distilled models (DistilBERT, DistilBERT SQuAD and tinyRoBERTa) performed extremely well for their smaller model size when compared to the state-of-the-art reference models BERT, XLNet, RoBERTa and DeBERTa. The most noticeable smaller model being tinyRoBERTa being

in the top three metric performers in EM and F1, while having 81.5 million parameters (or 25.4% less parameters than SciBERT and 55.7% less parameters than DeBERTa).

TABLE A.2: Summary of fine-tuned pre-trained transformer results - Validation and Test datasets

Model	no. params (trainable)	Val EM	Val F1	Test EM	Test F1	Start logits Acc	End logits Acc
Model	(trainable)	Val EM	Val F1	Test EM	Test F1	Acc	Acc
DistilBERT SQuAD	**65.2**	97.6078	98.6942	97.4433	98.7263	98.4532	98.4705
DistilBERT	66.4	96.2650	97.8671	96.3539	97.7532	97.9809	97.9686
tinyRoBERTa	**81.5**	**97.7323**	**98.9355**	**97.8880**	**99.0393**	98.4368	97.9930
BERT	107.7	97.0476	98.4156	97.2432	98.5154	98.0370	98.0961
SciBERT	**109.3**	**98.1058**	**99.1662**	**97.9546**	**99.0972**	**98.5453**	**98.4738**
XLNet	116.7	60.8804	79.1452	61.4940	79.7373	**98.5606**	**98.6418**
RoBERTa	124.1	97.0920	98.7026	97.1098	98.6710	98.0078	97.7810
DeBERTa v3	183.8	**98.3993**	**99.0854**	**98.3993**	**99.0579**	**98.8406**	**98.8357**

Transformer cross-validation (k=10, not shuffled) performance: The Table A.3 shows the results of various transformers and using a k-fold (k=10) cross-validation dataset. The four selected models RoBERTa, tinyRoberta, DistilBERT and SciBERT when fine-tuned and evaluated, all performed very well.
DistilBERT had the lowest k-fold cross-validation mean of 98.751% and tinyRoBERTa had the best cross-validation mean of 99.024%. It is further noted that tinyRoBERTa had the largest standard deviation of 0.166% and SciBERT the lowest standard deviation of 0.173%.
Model evaluation on the test data that was held-back reported similar high numbers. DistilBERT still had the lowest F1 of the four selected models, but RoBERTa (99.050%) took the lead over tinyRoBERTa (99.006%) by a narrow margin. Evaluating the k-fold means, which summarises the models skills, as being similar magnitude to the test F1 scores of Table A.2 helps support the F1 scores to be more trustworthy - since we expect the k-fold mean to be a less but similar in magnitude to the original individual test F1 score.

Transformer cross-validation (k=6, shuffled) performance: The Table A.4 similarly complements the results of RoBERTa, tinyRoberta, DistilBERT and SciBERT. A five fold split of all the data, and validation data was now shuffled and selected in the ratio of 85:15 for training and validation. We notice that the average validation F1 scores differed where tinyRoBERTa took the lead marginally with F1 score mean of 99.130% compared to RoBERTa's F1 score mean of 99.118%; and a position switch where DistilBERT on average perform slightly better than SciBERT by 0.02%. Again, the shuffled five fold cross-validation results are similar magnitudes to the original individual test F1 score in Table A.2.

Table A.3: Summary of selected pre-trained transformers - k-fold (k=10) CV datasets

validation	RoBERTa	tinyRoBERTa	DistilBERT	SciBERT
0	98.7845	99.0949	98.6450	98.9855
1	99.2035	98.3820	98.7330	98.7990
2	98.9054	99.2590	99.0130	99.2740
3	98.9465	99.0700	98.7423	99.1291
4	98.9424	98.8663	98.5151	98.7781
5	99.1249	99.0380	98.7510	98.8376
6	99.2811	99.3050	98.9714	98.7663
7	99.2506	99.2990	98.9527	98.9168
8	98.7747	99.0966	98.8078	98.7461
9	98.6201	98.8301	98.3762	98.7251
Val. Ave.	98.983	99.024	98.751	98.896
test	RoBERTa	tinyRoBERTa	DistilBERT	SciBERT
0	98.8820	99.0729	98.5830	99.0181
1	99.1141	98.5230	98.8800	98.8810
2	98.9363	99.1510	98.8220	98.9790
3	99.2026	99.0780	98.7689	98.8752
4	98.9710	99.0138	98.4156	98.9114
5	99.0967	99.0229	98.6730	98.9051
6	99.2403	99.0870	98.8512	98.7573
7	99.1669	99.0040	98.8593	98.8512
8	98.9527	99.0357	98.8170	98.6205
9	98.9338	99.0730	98.5594	98.9550
Test Ave.	99.050	99.006	98.723	98.875

Table A.4: Summary of selected pre-trained transformers - k-fold (k=5) CV data, shuffled

validation	RoBERTa	tinyRoBERTa	DistilBERT	SciBERT
1	99.03287	99.15260	98.81419	98.83564
2	99.20333	99.16132	98.82573	98.64917
3	98.94378	98.85993	98.93045	98.75544
4	99.25672	99.12143	98.87165	99.01265
5	99.15219	99.35531	98.99970	99.06396
Val. Ave.	99.118	99.130	98.888	98.863

Appendix B

Predictions

In this appendix, prediction information is given. Two tabled figures list random samples of predicted outputs in descending order of prediction score. The features tabulated are the model name, the question and context, the ground truth answer, the predicted answer, and predicted score. In addition, the context and the question was fed into ChatGPT 3.5 where the responses are reported. Note that all the ChatGPT 3.5 outputs where correct from a human evaluation, but from a computers perspective some of the responses returned hyphenated tokens from the context that originally did not have a hyphen.

Figure B.1 tabulates 14 random outputs of SciAD fine-tuned transformers that predicted correctly. Figure B.2 continues with the next 16 random outputs that predicted poorly from the 17^{th} output.

We observe a sample of the zero-shot outputs of SciAD models applies to the MMM domain where, for example tinyRoBERTa, had an exact match score of 16% of the 50 MMM examples, and it F1 score was calculated to be 49.2000%. Compared to RoBERTa that seems to have more exact matches of 24% of the 50 MMM examples, and a lead of F1 score to 52.20%.

Figure B.3 shows a selection of 8 MMM predicted outputs. The F1 scores are in decreasing order. The top two performers were RoBERTa and tinyRoBERTa which have better F1 scores than SciBERT and XLNet. Two of the eight had a ground truth of 'process value', RoBERTa had an exact match in both cases, whereas tinyRoBERTa partially predicted the first token 'process', and missed the second token 'value'. SciBERT got one exact match and the second one it predicted three tokens instead and meaning it added a token for whatever reason. XLNet for both cases only predicted the first token 'process' in 'process value'. For ground truth 'differential pressure' and the question 'what is PD?' we note the acronym letters are reversed, yet most of the models predicted correct expansions. SciBERT got confused the most and over reported four tokens 'differential pressure input warning'. Contrasting to the ground truth 'specific energy consumption' where RoBERTa, tinyRoBERTa, and XLNet missed understood the question, and only SciBERT successfully reported an exact match. The use of temperature and related to the acronym TT seemed to confuse the models in general, but tinyRoBERTa seemed handle to disambiguation the best.

	Model	Question	Ground truth	Predicted	Predicted Score	ChatGPT 3.5 Prediction	Context
1	shaun-e-j/allenai__scibert_scivocab_cased__SDU_AAAI	what is GSR?	group sparsity residual	group sparsity residual	1.0000	GSR stands for "Group Sparsity Residual."	It can be seen that the proposed group sparsity residual unk NLS uses less computation time than the competing methods except for BM3D unk EPLL and PGPD unk
2	shaun-e-j/microsoft__deberta-v3-base__SDU_AAAI	what is RNN?	random neural networks	random neural networks	1.0000	In the provided text, the acronym "RNN" stands for "Recurrent Neural Network."	After that unk the random neural networks can be used in real unk time unk
3	shaun-e-j/microsoft__deberta-v3-base__SDU_AAAI	what is OCC?	one class classifier	one class classifier	1.0000	OCC expands to "One-Class Classifier."	ADEPOS with B unk one class classifier enables saving in energy unk
4	shaun-e-j/distilbert-base-cased-distilled-squad__SDU_AAAI	what is RR?	reverse reachable	reverse reachable	1.0000	In the given text, RR stands for "Reverse Reachable."	Then we generate reverse reachable sets for each node in unk
5	shaun-e-j/allenai__scibert_scivocab_cased__SDU_AAAI	what is ESE?	extract similar entities	extract similar entities	0.9999	In the provided text, the acronym "ESE" stands for: Entity Similarity Extraction	Our approach differs from the above in that we perform extract similar entities by learning the semantics and relevant context to retrieve similar entities by analogy unk to accelerate the learning rate unk
6	shaun-e-j/deepset__tinyroberta-squad2__SDU_AAAI	what is IR?	information retrieval	information retrieval	0.9999	In the provided text, "IR" is an acronym for "Information Retrieval."	BugLocator unk one of the well cited information retrieval unk based techniques unk returns such results at the unk and unk positions respectively for the noisy and poor queries which are far from ideal unk
7	shaun-e-j/distilbert-base-uncased__SDU_AAAI	what is CA?	coded aperture	coded aperture	0.9999	In the provided text, the acronym "CA" stands for "Coded Aperture."	The central video is cropped and used to compute coded aperture and MS unk TRS at different scales as described in the testing protocol unk
8	shaun-e-j/deepset__tinyroberta-squad2__SDU_AAAI	what is CNN?	convolutional neural network	convolutional neural network	0.9998	CNN stands for Convolutional Neural Network.	Indeed unk deploying this baseline convolutional neural network detector over a bioacoustic sensor network will likely lead to a systematic underestimation of vocal activity of migratory birds at dusk and an overestimation at dawn unk
9	shaun-e-j/roberta-base__SDU_AAAI	what is RF?	radio frequency	radio frequency	0.9997	RF stands for "Radio Frequency."	Along with both protocols unk three approaches can be employed for the renewable and radio frequency energies and transmission management unk
10	shaun-e-j/bert-base-cased__SDU_AAAI	what is RNN?	recurrent neural network	recurrent neural network	0.9987	In the given text, "RNN" is an acronym that stands for "Recurrent Neural Network."	In fact unk a LSTM layer has more than four times parameters than a vanila unk recurrent neural network layer unk
11	shaun-e-j/bert-base-cased__SDU_AAAI	what is TDS?	taint dependency sequences	taint dependency sequences	0.9949	In the provided text, the acronym "TDS" expands to "Taint Dependency Sequences."	The reason for doing so is that smaller taint dependency sequences may be very trivial or may provide less precise path unk
12	shaun-e-j/distilbert-base-cased-distilled-squad__SDU_AAAI	what is DC?	datacenter	datacenter	0.9888	DC stands for "Datacenter."	Along with evolving improvement in cloud computing and their advanced virtualization techniques related to network function virtualization unk many papers have been focused on cost saving under better utilization of computing resources in cloud unk based mobile core networks with the objective of finding the optimal placement of VNFs within the same datacenter unk
13	distilbert-base-cased-distilled-squad	what is FEC?	forward error correction	forward error correction block	0.7810	FEC stands for "Forward Error Correction."	This forward error correction block can be adjusted in real - time depending on Markov models to estimate the PLR and the number of continuous losses , to boost video transmissions .
14	shaun-e-j/distilbert-base-uncased__SDU_AAAI	what is EI?	expected improvement	expected improvement	0.6224	EI stands for "Expected Improvement."	We can consider the expected improvement to assess unk unk

FIGURE B.1: Random Samples of SciAD fine-tuned Transformer Outputs.
Descending order of strong prediction scores.
The respective fine-tuned SciAD models predicted correctly.
The ChatGPT experiment return all the "correct" expansions; and example 3, returned hyphened token "One-Class" from the context containing "one class"

	Model	Question	Ground truth	Predicted	Predicted Score	ChatGPT 3.5 Prediction	Context
15	shaun-e-j/ xlnet-base-cased ___SDU_AAAI	what is MT?	machine translation	machine translation	0.5570	MT stands for "Machine Translation."	D17 - 123 and isabelle2018challenge prepared challenge sets for machine translation evaluation covering fine - grained phenomena at morpho - syntactic , syntactic , and lexical levels .
16	shaun-e-j/ roberta-base ___SDU_AAAI	what is MI?	myocardial infarction	myocardial infarction	0.5184	In the provided text, "MI" stands for "Myocardial Infarction."	The results of experimental setup depict that the proposed system has achieved an average accuracy of unk for PAC unk for myocardial infarction and unk for PVC unk with an average error rate of unk for PAC unk unk for myocardial infarction and unk for PVC on real electrocardiogram datasets including Physionet and European ST unk T Database unk EDB unk unk
17	distilbert-base-cased-distilled-squad	what is SU?	secondary user	a time fraction	0.3184	In the provided text, the acronym "SU" stands for "Secondary User."	Accordingly , each secondary user submits its bid including (i) the PJ power , (ii)
18	shaun-e-j/ xlnet-base-cased ___SDU_AAAI	what is EM?	exact match		0.0381	In the text you provided, "EM" stands for "Exact Match."	Similar phenomenon is observed on exact match .
19	microsoft/ deberta-v3-base	what is DBN?	deep belief network	competing algorithms .	0.0125	DBN stands for "Deep Belief Network."	ECS - deep belief network outperforms all other competing algorithms .
20	allenai/ scibert_scivocab_cased	what is ML?	machine learning	is their inability	0.0096	In the text, ML stands for "Machine Learning."	The challenge of machine learning and AI methods is their inability to take such intuitive leaps .
21	xlnet-base-cased	what is ES?	energy storage		0.0071	ES stands for "Energy Storage."	Global Dynamics and Stability Analysis presents the block digram of the feedback loop for each of the distributed energy storage controllers .
22	xlnet-base-cased	what is IR?	influence ratio		0.0065	IR stands for "Influence Ratio."	Figure Ratioinfluence ratio visualizes the relationship among ratio , influence ratio , and elapsed time from the last comment .
23	bert-base-cased	what is PCA?	principal component analysis	Note , principal component analysis provides two important features that we will use in	0.0052	PCA stands for Principal Component Analysis.	Note , principal component analysis provides two important features that we will use in the sequel .
24	allenai/ scibert_scivocab_cased	what is MF?	matrix factorization	Interestingly , the optimally terminated NCF models actually suffered	0.0039	MF stands for Matrix Factorization.	Interestingly , the optimally terminated NCF models actually suffered from less overfititng than the matrix factorization models did at higher parameter counts .
25	microsoft/ deberta-v3-base	what is SU?	secondary user	a threshold to improve the accuracy of	0.0029	In the provided text, the acronym "SU" stands for "Secondary User."	The secondary user then combines its own sensing result with the results of other secondary users which have reputation values greater than a threshold to improve the accuracy of sensing .
26	bert-base-cased	what is RS?	relay station	station is of mitigating	0.0017	In the provided text, "RS" is an acronym that stands for "Relay Station."	When , the slope is less than the other cases showing that the higher the is , the less capable relay station is of mitigating the SI .
27	distilbert-base-uncased	what is OCT?	odd cycle transversal	include a MIB	0.0010	In this text, OCT stands for "Odd Cycle Transversal."	To show the correctness of , we must show that for every MIB In , we include a MIB in which includes all of its non - odd cycle transversal nodes and a node
28	distilbert-base-uncased	what is PNN?	product based neural network	can firstly train a part of product based neural	0.0009	In this text, "PNN" expands to "Product-based Neural Network."	Inspired by Net2Net , we can firstly train a part of product based neural network (e.g. , the FNN or FM part) as the initialization , and then start to let the back
29	deepset/ tinyroberta-squad2	what is LOD?	level of detail	finer details can not be represented or recovered	0.0000	LOD stands for "Level of Detail."	While some level of detail techniques can be applied to generate different terrain resolutions , finer details can not be represented or recovered .
30	deepset/ tinyroberta-squad2	what is RNN?	random neural networks	random neural networks	0.0000	In the given text, the acronym "RNN" expands to "Random Neural Networks."	In the same way as in the other mechanisms , the Offline process is responsible for the random neural networks training and validation steps .

FIGURE B.2: Random Samples of SciAD fine-tuned Transformer Outputs.
Descending order of weak prediction scores.
The respective fine-tuned SciAD models predicted poorly.
The ChatGPT experiment return all the "correct" expansions; and example 28, similarly returned hyphened token "Product-based" from the context containing "product based"

Context	Question	Ground truth	roberta-base	tinyRoBERTa
Portable Water Reservoir Inlet 2 Flow process value	What is PV?	process value	process value	process
Grey Water Supply Flow process value	What is PV?	process value	**process value**	process
Gland Service Water Flow From Tank 999-Tk-001 process value	What is PV?	process value	**process value**	**process value**
Granulator Make-Up Water Tank Tk-001 Level process value	What is PV?	process value	**process value**	process
SXYZ Storage Silo 999-Sf-001 Level Simulated	What is SIM?	Simulated	**Simulated**	**Simulated**
PVR Fan Bearing Oil Supply - Temperature High Setpoint	What is TT?	Temperature	Setpoint	point
PVR Fan Bearing Oil Supply - Temperature Output	What is TT?	Temperature	**Temperature**	**Temperature**
Fan bearing temperature motor side axial High Setpoint	What is TT?	Temperature	**temperature**	temperature motor side axial
Fan bearing temperature motor side axial Output	What is TT?	Temperature	**temperature**	temperature motor
Crystalliser PVR Fan 2 Hydraulic Oil Temperature High Setpoint	What is TT?	Temperature	Setpoint	Temperature High Setpoint
Crystalliser PVR Fan 2 Hydraulic Oil Temperature Output	What is TT?	Temperature	Oil Temperature	**Temperature**
Cyclone 1 differential pressure Input warning high	What is PD?	differential pressure	**differential pressure**	**differential pressure**
	exact match	100	24	16
	f1	100	52.20	49.20

scibert_scivocab_cased	bert-base-cased	xlnet-base-cased
Portable Water Reservoir	**process value**	process
process	**process value**	<empty>
process value	**process value**	process
process value	**process value**	process
SXYZ Storage	Silo	SXYZ Storage
Temperature High Setpoint	Fan Bearing Oil Supply	<empty>
Temperature Output	Fan Bearing Oil Supply	<empty>
temperature motor	motor side axial	**temperature** motor side
temperature motor	motor side axial	**temperature** motor side
Temperature High	Hydraulic	Crystalliser
Temperature Output	Hydraulic	Crystalliser
differential pressure Input warning	**differential pressure**	**differential pressure**
10	14	6
46.78	38.80	23.00

FIGURE B.3: Selection of MMM data are shown.
Predicted outputs of the SciAD fine-tuned Transformer are compared for MMM data.
The F1 scores are in decreasing order.
RoBERTa and tinyRoBERTa have better F1 scores than SciBERT and XLNet.
Where respectively (F1=52.20% and F1=49.20%) > (46.78% or 23.00%)

Appendix C

Modelling

C.1 Model Fit

We describe the model fit results for two machine learning methods. End-to-End Memory LSTM method is discussed in terms of its measured model accuracy, model F1 score, model loss, and the model learning rate. An example transformers, RoBERTa and XLNet, are discussed by their model loss and start logits accuracy.

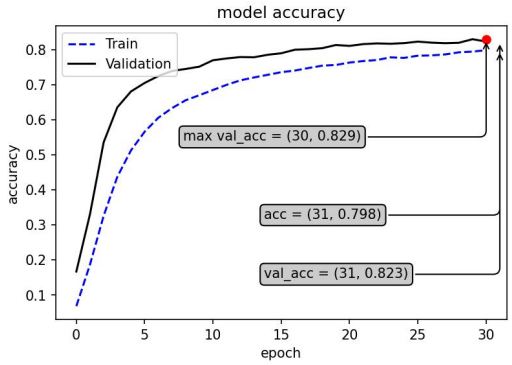
C.1.1 End-to-End Memory LSTM

Table 4.1 reported an adequate **78.75% baseline metric** (for held-back data) for weighted F1 score. We show that the training metrics, for example model loss, over the 31 epochs that the validation results "tracked" the profile of the training results - there was an intersection of training and validation values around the 26^{th} epoch, and the final gap between training and validation was not large and training was stopped early.

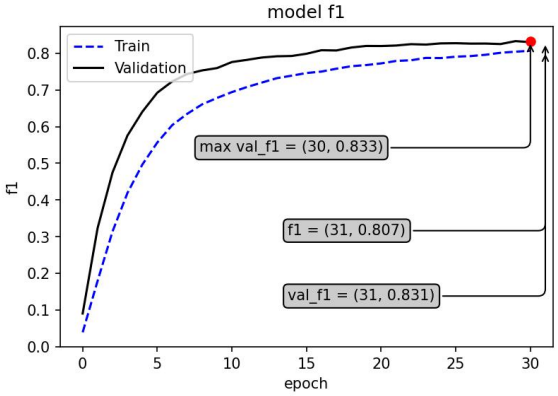
In Figure C.1 (a) the accuracy of the validation data throughout the training was above the training model accuracy. The max validation accuracy achieved was 0.829 on the 30th epoch. Figure C.1 (b) the best validation F1 score of 0.8332 (or 83.32%).

In Figure C.2 (a) shows the model loss over the trained 31 epochs. Majority of the learning occurred by the 17th epoch where the model loss was 0.9893 at a learning rate of 0.0019; and thereafter incremental learning improvements still occurring and lowest validation data set loss of 0.834 achieved at the 26th epoch.

Figure C.2 (b) shows the model learning rate that occurred every during the training, the learning rate was a calculated schedule that decayed by 7.5% approximately every 7 epochs. The initial learning rate was 0.002 and step down four times and finished on 0.0018.

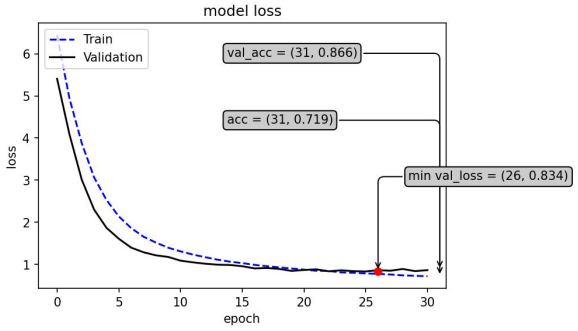


(A) Accuracy. Max validation accuracy 0.829.

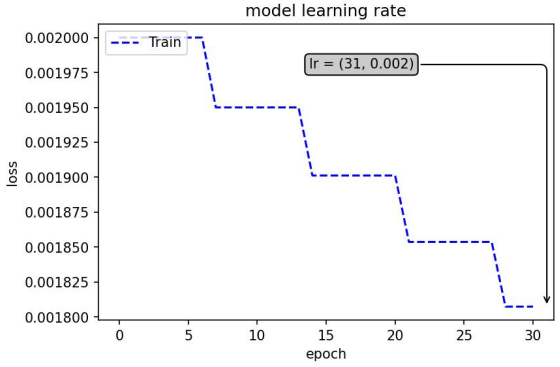


(B) F1 score. Achieved validation f1=83.30% (and test data f1=78.75%

FIGURE C.1: Baseline (LSTM based) Model Training Accuracy and F1 score



(A) Loss of 0.834 at 26th epoch. Model Accuracy of 0.719.



(B) Learning rate (calculated). Initial value of 0.002

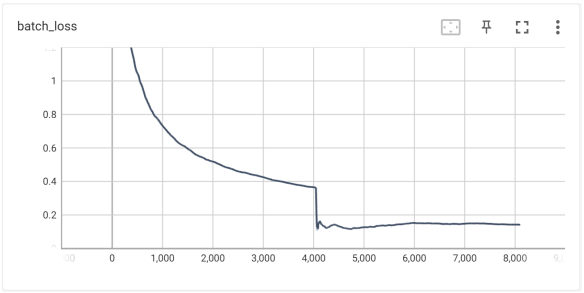
FIGURE C.2: Baseline (LSTM based) Model Training Loss and Learning rate

C.1.2 Transformers

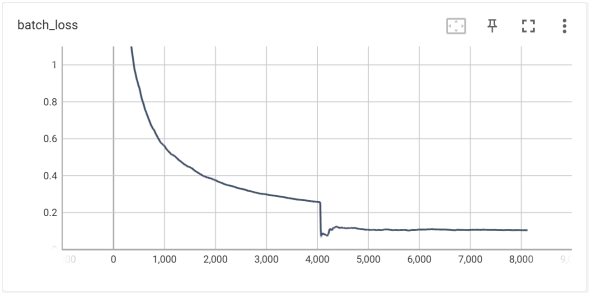
The training profiles for assessed transformers in terms of iterations all showed similar batch profiles for the two epoch runs. Note the step change at the epoch 1 to epoch 2 transition that occurs and is not fully understood by the author.

In Figure C.3 shows the batch loss profiles after the 2nd epoch or batch step 8100, (a) RoBERTa model loss was 0.1422 and (b) XLNet model loss was 0.1040.

Figure C.4 shows the batch start logits accuracy profiles after the 2nd epoch or batch step 8100, (a) RoBERTa model start logit accuracy was 0.9801 and (b) XLNet model start logit accuracy was 0.9856.

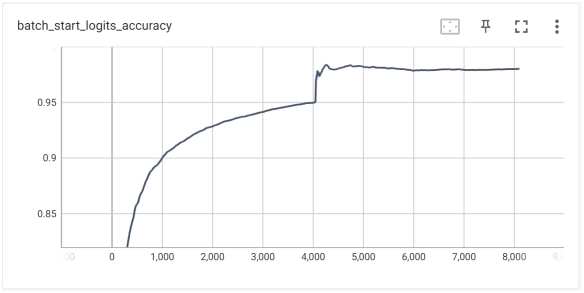


(A) RoBERTa
Loss of 0.1422
batch step 8110

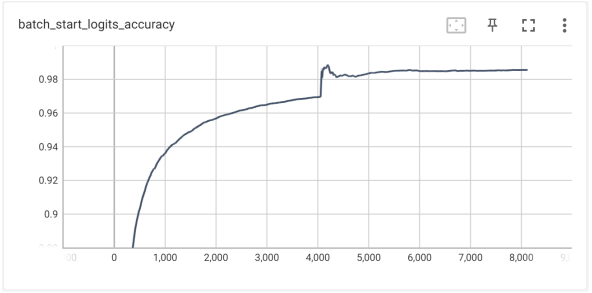


(B) XLNet
Loss of 0.1040
batch step 8130

FIGURE C.3: Transformer Fine-tuning - TensorBoard Batch Loss



(A) RoBERTa
Start Logits Accuracy of 0.9801
batch step 8110



(B) XLNet
Start Logits Accuracy of 0.9856
batch step 8130

FIGURE C.4: Transformer Fine-tuning - TensorBoard Batch Start Logits Accuracy

C.2 Code

C.2.1 Requirements

The (Python) Framework versions utilised were:

- TensorFlow[1] 2.6.0
- Transformers[2] 4.28.1
- Tokenizers 0.13.3
- Datasets 2.11.0

C.2.2 Hardware

- NVIDIA GeForce RTX 4070 Ti, Driver Version: 531.61, CUDA[3] Version: 12.1
- Google Colaboratory[4]

[1] TensorFlow is an end-to-end open source platform for machine learning
[2] Transformers provides APIs and tools to easily download and train state-of-the-art pretrained models
[3] The NVIDIA CUDA® Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks.
[4] Colab is a hosted Jupyter Notebook service. Colab is especially well suited to machine learning, data science, and education.