ABB-BERT: A BERT model for disambiguating abbreviations and contractions

Prateek Kacker[1]*, **Andi Cupallari**[1], **Aswin Gridhar Subramanian**[2†] and **Nimit Jain**[1]
Novartis Pharmaceuticals, NJ, USA[1]
School of Informatics, University of Edinburgh[2]

Abstract

Abbreviations and contractions are commonly found in text across different domains. For example, doctors' notes contain many contractions that can be personalized based on their choices. Existing spelling correction models are not suitable to handle expansions because of many reductions of characters in words. In this work, we propose ABB-BERT, a BERT-based model, which deals with an ambiguous language containing abbreviations and contractions. ABB-BERT can rank them from thousands of options and is designed for scale. It is trained on Wikipedia text, and the algorithm allows it to be fine-tuned with little compute to get better performance for a domain or person. We are publicly releasing the training dataset for abbreviations and contractions derived from Wikipedia.

1 Introduction

We use abbreviations and contractions (called "short forms" in this paper) while quickly typing on digital apps. They are used to save time or effort in typing and may be unique to us; therefore, sometimes, only we can understand them. There is no reliable dictionary of short forms to be referred because it can be specific to a context or a person. The short forms often have multiple meanings depending on the domain or the person. In Table 1, consider the sentence *"The doctor saw an AS cd at tl"* written in a notepad by sales representative at pharmaceutical company or local news reporter in Las Vegas. It may be a shorthand for *"The doctor saw an Ankylosing Spondylitis candidate at trial"* for the sales representative or *"The doctor saw an Ace of Spade card at the table"* for the news reporter. Applying downstream NLP AI Algorithms to this shorthand text gives poor performance because they have not been trained on personalized shorthand text. To build better AI systems, we should expand the short forms in the sentences for the domain or the user before using it downstream. Since there is no correct answer for expansions and numerous right choices based on the domain and the user, ranking is the better way to handle short forms text in real-world AI applications.

Text notes	
Notes 1	The doctor saw AS cd at tl
Notes 2	The doctor saw AS cd at tl
Ground Truth	
Notes 1	The doctor saw Ankylosing Spondylitis candidate at trial
Notes 2	The doctor saw Ace of Spades card at the table
ABB-BERT input	
Notes 1	The doctor saw at *[ABB] [ABB]* at *[ABB]*
Notes 2	The doctor saw at *[ABB] [ABB]* at *[ABB]*
ABB-BERT outputs (sorted list on rank)	
Notes 1	*[ABB]*- [Ankylosing Spondylitis, ...] *[ABB]*- [candidate, ...] *[ABB]*-[trial,...]
Notes 2	*[ABB]*-[Ace of Spades,...] *[ABB]*-[card,...] *[ABB]*-[table,...]

Table 1: Text notes made by one can be ambiguous for others. **Notes 1** was written by pharmaceutical sales, and **Notes 2** was written by local news in Las Vegas. **ABB-BERT** can suggest the best replacement using a fine-tuned model for a domain

The definition of abbreviation is simple, and everyone understands it. For example, *USA* stands for the United States of America, or *MS* stands for Multiple Sclerosis. On the other hand, a contraction is a misspelling or shortening of any word, such as *dr, drs, dctr* etc., for a doctor or *ptnt, pnt, pt* etc., for a patient. Current spelling correction models fail to capture the correct form for all possible scenarios

* Correspondence to: prateek.kacker@novartis.com
† Part of work was done during employment at Novartis

Proceedings of the 18th International Conference on Natural Language Processing, pages 289–297
Silchar, India. December 16 - 19, 2021.

because of the many reductions of characters in short forms.

Large NLU language models like BERT (Devlin et al., 2019a), RoBERTa (Liu et al., 2019), ALBERT (Lan et al., 2020), or DeBERTa (He et al., 2020) are trained on normalized data from different domains but not with personalized or domain-specific short forms and hence reduce the model performance in downstream NLP tasks. For example, in a classification task, the contractions or abbreviation might be critical in determining the class and can lead to a wrong classification (false positive or false negative). To solve this problem, ABB-BERT can normalize the text by ranking the options to find the best choice for abbreviation or contraction, leading to better downstream performance.

In the past, much work has been done on normalizing text data. Misspellings (simple spelling mistakes) have been handled well by AI models. Recent work by Tan et al. (2020) introduced TNT, a model that was developed to learn language representation by training transformers to reconstruct text from operation types typically seen in text manipulation, which they show is a potential approach to misspelling correction. Another AI algorithm Neuspell (Jayanthi et al., 2020) is a spelling correction toolkit that captures the context around the misspelled words. We have noticed that misspelling AI models do not perform well with contractions because of loss in information due to contraction and the number of possible variations for the right choice based on domain.

Kreuzthaler et al. (2016) introduced a data-driven statistical approach and a dictionary-based method for the task of abbreviation detection. They show some success of these approaches; however, as their approach depends on a dictionary with a limited number of entries, it cannot be scaled or extended to other domains. Joopudi et al. (2018) trained Convolutional Neural Network (CNN) models to disambiguate abbreviation senses and found a 1–4 percentage points higher performance for CNN compared to Support Vector Machines. These results were robust across different datasets.

Li et al. (2019) showed that topic modeling combined with attention networks could help get better results on abbreviation disambiguation because topics provide the context for the neural networks. To improve the performance on bio-medical data, Jin et al. (2019b) utilized pre-trained model BioELMO

Algorithm 1 *contraction*

Input: word

Output:List of possible contractions

1: Remove any other characters except $a-z$,$A-Z$ and lower case the word
2: Remove all the vowels a, e, i, o, u
3: Select all characters except 1^{st} character
4: Find all possible combinations of selected characters without changing order
5: Append the first character to each item in the list
6: Return list

Algorithm 2 *abbreviation*

Input:sentence

Output:List of tuples (Abbreviations,expansions)

1: Identify capitalized word in sentence and their location
2: Identify capitalized word sequences with length two or more
3: If two sequences are seperated by prepositions or conjuctions then connect them to form a sequence
4: Create a list of tuples (initals of uppercase words in sequence, sequence)
5: Return list

(Jin et al., 2019a) which gets better-contextualized features of words. Then the features are fed into abbreviation-specific bidirectional LSTMs where the hidden states of the ambiguous abbreviations are used to assign the exact definitions. Recently, Pan et al. (2021) proved that BERT-based algorithms combined with training strategies like dynamic negative sample selection and adversarial training are very effective in Scientific AI domain acronym disambiguation datasets (SciAD) (Veyseh et al., 2020).

The contribution of this paper is threefold. First, we propose ABB-BERT which uses a ranking algorithm by combining BERT (Devlin et al., 2019b) and architecture by LaBSE in Feng et al. (2020) on short forms options based on context. We introduce a new token *[ABB]* that replaces all short forms. Second, we show that ABB-BERT is a practical and scalable way to deal with un-normalized text across domains. Third, we publicly release the dataset and code [1] for ABB-BERT for future work.

[1]Dataset and code available at https://github.com/prateekkacker/ABB-BERT

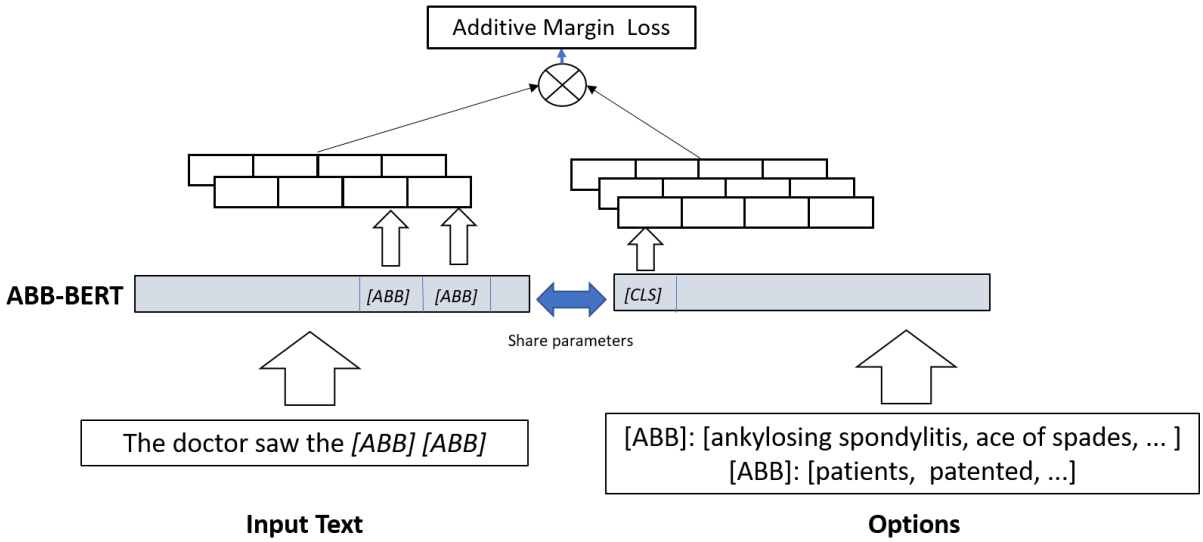

Figure 1: Graphical representation of the training and inference on ABB-BERT. The sentence written by the sales rep is *"The doctor saw the AS ptnt"* but for training, the sentence is modified to *"The doctor saw the [ABB] [ABB]"*. The model is trained to minimize the additive softmax loss of *[ABB]* corresponding to *AS* and *ptnts*. The ground truth for the above example is *"The doctor saw the Ankylosing Spondylitis patient"*. During inference, ABB-BERT ranks the several options given per *[ABB]*. ABB-BERT can be fine-tuned to improve the performance of a domain.

Key	Value	Number of choices
ptnt	patient, patent, potent, potential ...	4736
dctr	doctor, director, documentary, declaratory ...	2555
tl	table, trial, tool, tuberculosis ...	81236

Table 2: Selected examples of key-value pairs in *dict_cont*.

Key	Value	Number of choices
as	Ankylosing Spondylitis, Ace of Spades, Astronomy and Space ...	89119
acl	Association for Computational Linguistics, Avant Co. Ltd., Albany Club in London ...	2259
usa	United States of America, Urban Songwriter Award, Ultimate Sports Adventure ...	1608

Table 3: Selected examples of key-value pairs in *dict_abb*.

2 ABB-BERT

The goal of the algorithm is to rank the options for short forms. As shown in 1, the input sentence X may contain one or more short forms. We assume that short forms' location in the sentence and the character composition is known for training purpose. A typical example of sentence with short form can be seen in Table 4. We substitute the short forms with $[ABB]$ and the algorithm uses character composition to get several options for short forms, create embeddings and calculate scores to rank each option.

Original Sentence	With contractions and abbreviations
When I got to the house, **Mrs. Everett**, the **housekeeper**, told me that Hermione was in her room, watching her maid pack.	**WI** got to the house, **ME**, the **hs** told me that Hermione was in her room, watching her maid pack.
The Sydney area has been **inhabited** by **indigenous** Australians for at least 30,000 years.	The Sydney area has been **id** by **ig** Australians for at least 30,000 years
Bosnian **claims** of Serbian annexation attempts in 1993 were brought to the **World Court**.	Bosnian **cs** of Serbian annexation attempts in 1993 were brought to the **wc**.

Table 4: Selected examples of GLUE Benchmark datasets. They have been manually edited to create training data for ABB-BERT

2.1 Lookup Tables for Options

ABB-BERT ranks options based on thousands of choices for expansions for short forms present in the lookup tables *dict_cont* and *dict_abb*. To create

these tables, English Wikipedia has been parsed for words for contractions and full forms for abbreviations using the rule-based methods in Algorithm 1 and Algorithm 2, respectively. After observing thousands of short forms used in real-world datasets, these rules were created, which we cannot share publicly. Using these rules, we created key-value pairs for lookup tables, *dict_cont* and *dict_abb*, from words and abbreviations extracted from English Wikipedia. Given an abbreviation or contraction, these lookup tables list words that can be the possible expansion. We can see the output of Table 2 and Table 3, and this list can be huge; hence scalability is crucial for ABB-BERT.

2.2 Model

ABB-BERT is based on the BERT architecture. In order to make it lightweight, it is pre-trained on an uncased BERT base model. ABB-BERT requires that every contraction and abbreviation be replaced with *[ABB]* token. Since *[ABB]* is not present in the default vocabulary of BERT, the vocabulary has to be modified to include this special token. Consider a sentence X in dataset D. After the tokenization of X, a sequence of tokens $(x_1, x_2, ...x_n)$ is generated. In this setup, x_1 is the *[CLS]* token for every sentence. We have already replaced the short forms with the *[ABB]* tokens hence we know their exact locations. For simplicity, let $(x_a, x_b, ...)$ represent the *[ABB]* token corresponding to indices $I = (a, b...)$. The BERT output $\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_n}$ corresponding to each token $x_1, x_2, .., x_n$ can be represented as

$$\boldsymbol{z_1}, \boldsymbol{z_2}, ..., \boldsymbol{z_n} = BERT(x_1, x_2, ..., x_n) \quad (1)$$

On top of the BERT model, there is a feed-forward neural network $f(.)$. The output vectors from BERT, $\boldsymbol{z_i}$, go through this neural network such that $\boldsymbol{y_i} = f(\boldsymbol{z_i})$. The final combined representation of the output is

$$\boldsymbol{y_1}, \boldsymbol{y_2}, ..., \boldsymbol{y_n} = \boldsymbol{ABB\_BERT}(x_1, x_2, ..., x_n) \quad (2)$$

$\boldsymbol{y_1}$ is the corresponding output for always the token representing from *[CLS]* and $\boldsymbol{y_a}, \boldsymbol{y_b}$... for *[ABB]* because of the indices I.

We do similar exercise for short forms. Each short form at *[ABB]* can have thousands of options and can be found from *dict_cont* and *dict_abb* tables. For location a, the options are denoted by $\boldsymbol{S_a}$ which is list of options $[S_a^1, S_a^2, ..., S_a^{o_a}]$ and length of the list is denoted by o_a. The tokenizer converts S_a^1, the first option to $(s_a^{1,1}, s_a^{1,2}, ..., s_a^{1,n})$ and similarly for other options $S_a^2, ..., S_a^{o_a}$. Similar to X, every option S_a is propogated through $\boldsymbol{ABB\_BERT}$. The output is represented for S_a^1 is represented as:

$$(\boldsymbol{t_a^{1,1}}, \boldsymbol{t_a^{1,2}}, ..., \boldsymbol{t_a^{1,n}}) = \boldsymbol{ABB\_BERT}(s_a^{1,1}, s_a^{1,2}, ..., s_a^{1,n}) \quad (3)$$

The equation 3 is applied to other options in $\boldsymbol{S_a}$ also. For options, there will not be any *[ABB]*. *[CLS]* is the first and the only relevant token hence the notations can be simplified by dropping the location information. For instance, $s_a^{1,1}$ to s_a^1, $s_a^{2,1}$ to s_a^2 etc and similarly for $\boldsymbol{t_a^{1,1}}$ to $\boldsymbol{t_a^1}$, $\boldsymbol{t_a^{2,1}}$ to $\boldsymbol{t_a^2}$ etc. The algorithm uses additive margin softmax loss, discussed in the next section, to rank the outcomes.

2.3 Dual Encoder with Additive Margin Softmax Loss

The architecture of ABB-BERT with additive margin softmax loss is shown in figure 1. The architecture is similar to the one used by Feng et al. (2020). We use dual-encoders which feeds a scoring function and determines the rank of the alternatives based on the cosine similarity measure, and hence such models are well suited for ranking problems. We use the additive margin softmax loss function introduced in Wang et al. (2018b). Later on, Yang et al. (2019) used a slightly modified version of this loss, and Feng et al. (2020) used it in their language-agnostic LABSE model.

For this paper, the short forms disambiguation problem is modeled as a ranking problem to find the best option S_a for short form at index a in sentence X where S_a is one of the alternatives in $[S_a^1, S_a^2, ..., S_a^{o_a}]$. The ranking of the options is evaluated by the cosine similarity score ϕ . For ABB-BERT, ϕ scores for all the options at location a, is calculated by calculating cosine similarity ϕ between $\boldsymbol{y_a}$ and $\boldsymbol{t_a^1}, \boldsymbol{t_a^2}, ..., \boldsymbol{t_a^{o_a}}$ for options $S_a^1, S_a^2, ..., S_a^{o_a}$. Ranking of the options at location a is done by sorting ϕ scores.

To train the algorithm, we find the conditional probability $P(S|X)$ for options and for all locations. For example, S_a^1, the first option at location a will have $P(S_a^1|X)$ as:

$$P(S_a^1|X) = \frac{e^{\phi(t_a^1, y_a)}}{\sum_{i=1}^{o_a} e^{\phi(t_a^i, y_a)}} \quad (4)$$

This can be extrapolated to other options and other locations. For training purposes, for each location, the first option is the ground truth option.

	Metrics	**Results A**	**Results B**	**Results C**
COLA	Matthews corr.	52.6	22.8	**48.1**
SST2	acc	93.6	20.4	**92.9**
STS-B	Pearson/ Spearman corr	84.9/83.4	62.5/61.2	**75.0/73.8**
QQP	acc./F1	71.6/89.2	54.5/84.9	**65.4/88.3**
MNLI Matched	acc.	84.5	71.3	**77.9**
MNLI Mis-matched	acc.	83.4	72.0	**77.1**
MRPC	acc./F1	86.6/81.6	**79.8/75.4**	75.9/71.5
QNLI	acc.	90.9	**83.4**	82.6
RTE	acc.	64.4	56.7	**57.8**
WNLI	acc.	57.5	**61.0**	58.2

Table 5: Results of inference of downstream task trained on a single BERT-base-uncased model on GLUE Dataset on the respective tasks. Results A are obtained on the original test datasets. Results B are obtained on test datasets manually edited by introducing short forms. Results C are obtained on the test datasets improved by ABB-BERT by selecting 1^{st} option

Additive margin softmax extends the cosine similarity ϕ by introducing a large margin, m, only around correct option. The margin improves the separation between the correct option and other options. Moreover, we scale the cosine values using a hyper-parameter s in the equation 5. We select a large value, which accelerates and stabilizes the optimization (see (Wang et al., 2018b)). Equation 5 represents the loss function and is optimized during training.

Substituting for the new scoring functions, the objective loss function for single sentence X becomes:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=a,b,..}^{I} \sum_{o=1}^{n_i} \frac{e^{s(\phi(t_i^o, y_i) - m)}}{\sum_{k=1}^{n_i} e^{s(\phi(t_i^k, y_i) - m)}} \quad (5)$$

where

$$m = \begin{cases} 1 \geq m \geq \text{-}1 & o\text{=}1 \text{ or } k\text{=}1 \\ 0 & \text{otherwise} \end{cases}$$

$$s \gg 1$$

2.4 Scalable and personalized ABB-BERT

ABB-BERT might have to work real-time during inference in some applications to generate options for downstream tasks, though forward pass through BERT over thousands of alternatives can be expensive and time-consuming. Once ABB-BERT gets deployed, it is expected to get better in ranking for a domain, a user, or group of users with new annotations and training runs; hence there should be a personalization phase equivalent to finetuning the model for a person or a domain. In the *personalization phase*, it is not computationally possible to perform a forward pass on an entire list of options or retrain ABB-BERT again as the inference may be on a device with limited compute. To prepare for the *personalization phase* and to reduce the inference time, *dict_cont* and *dict_abb* is parsed for expansions for all possible options and ABB-

	% Correct outcomes (longest contr.)	**% Correct outcomes (short contr.)**
Wikipedia	63.7	11.2
Covid Dataset	61.7	11.0
Apps Review	54.4	0.0
US Bill	58.9	0.67
ECTHR	70.0	13.9

Table 6: Neuspell results on test sets of different domains on contractions. The model performs well with the longest contraction because it has the most number of words. The performance of Neuspell on abbreviations was close to 0 for all datasets.

BERT embeddings $\boldsymbol{t_i}$ are stored in a lookup table *dict_embed* for each expansion. The parameters for ABB-BERT and the table *dict_embed* are then frozen. ABB-BERT is modified by adding a single feed forward layer $g(.)$ parametrized by θ such that for embeddings y_i for sentences from equation 2 and embeddings $\boldsymbol{t_x^o}$ for options from equation 3, is modified. Training is done only on layer $g(.)$ to reduce training time.

$$\boldsymbol{u_i^o} = g(\boldsymbol{t_i^o}, \theta)$$

and $i \in I$ and for input sentences $\boldsymbol{y_i}$

$$\boldsymbol{z_i} = g(\boldsymbol{y_i}, \theta)$$

Here $\boldsymbol{u_i^o}$ and $\boldsymbol{z_i}$ replaces $\boldsymbol{t_i^o}$ and $\boldsymbol{y_i}$ respectively in equation 5. The model parameters are initialized by θ_0 such that $x = g(x, \theta_0)$. During the personalization phase, only parameters θ are trained which is not computationally expensive and can be done on the device. During inference, ABB-BERT does a forward pass only for input sentence $\boldsymbol{y_i}$. ABB-BERT does not need to do a forward pass on options and it can get embeddings $\boldsymbol{t_i^o}$ directly from *dict_embed* and a forward pass with parameters θ.

3 Experimental Setup

3.1 Data Preparation and Pre-training ABB-BERT

Training of ABB-BERT requires significant preparation of train, test, and validation data. We have taken English Wikipedia and extracted random sentences for datasets. We know that Wikipedia does not have contractions as it is a very clean dataset. Hence we had to create the datasets manually for ABB-BERT based on short forms in algorithm 1 and algorithm 2 respectively. For contractions, 15% of words in a sentence are selected at random. Using algorithm 1, a random contraction is selected to get options from *dict_cont*. Train, test and validation datasets contains >1M, >100K , >100k *[ABB]* tokens respectively. The ground truth, which is the correct expansion, is always the first word of the options in training, test, and validation datasets. In all the datasets, we have only 50 options per *[ABB]*. In real-world scenarios, there will be thousands of options. Pre-training of ABB-BERT was done on NVIDIA K80 GPUs for a week on Wikipedia training data with Adam optimizer and a lr equal to $5e-06$. After hyper parameter optimization, m was chosen to be 0.8, and s was 30.

3.2 Results

Each sentence in ABB-BERT can have multiple *[ABB]* and performance is calculated at each location at $I = (a, b, ...)$ There are two metrics relevant to this experiment. First is the average of rank (R) of the correct ground truth option, and second is average of difference (Dif) between cosine value (ϕ) of input sentence *[ABB]* & correct option which is the first option in training data and average cosine value of the input sentence $[ABB]$ & rest of the options

$$Dif = \phi(\boldsymbol{t_a^1}, \boldsymbol{y_i}) - (\sum_{n=2}^{o_a} \phi(\boldsymbol{t_a^n}, \boldsymbol{y_i}))/(o_a - 1) \quad (6)$$

We understand that the best average rank of the model outcome on the test set is 1, and Dif should be close to m on average. The larger the value of Dif, the better ABB-BERT is in predicting the outcome.

In the first experiment, we evaluate the impact of short forms on any downstream task. In order to model the impact, we took GLUE Benchmark (Wang et al., 2018a) tasks as a downstream task. Table 5 column **Results A** show the performance of the BERT-base-uncased model on each task without any changes to test data. We manually introduced short forms in test sets of each task using techniques mentioned in section 3.1. There is a marked reduction of performance in most datasets, as shown in table 5 in column **Results B**. Then we corrected each test set with ABB-BERT predictions selecting only the 1^{st} rank option from 50 options. The performance of the new test set is shown in the table 5 in column **Results C**. In the second experiment, we wanted to measure the model performance improvement after the personalization phase. Hence we tested it out on three domain datasets which were bio-medical, legal, and reviews datasets. For the biomedical domain, the Covid-19 QA dataset by Möller et al. (2020) was used. For the legal domain, US Legislation Corpus by Kornilova and Eidelman (2019) and European Court for Human Rights (ECTHR) database by Chalkidis et al. (2019) was used. For the technical domain, the Android Applications User Review dataset by Grano et al. (2017) was used. Sentences from paragraphs were extracted for train, validation, and test datasets. The lookup tables *dict_cont* and *dict_abb* were used to create short forms for all the datasets and parameters θ of $g(.)$

	Pre-personalization			Post-personalization					
	Avg. Rank over 50 options	[ABB] count	Avg. Diff	%(Top 3 ranks) in test set	Avg. Rank	Avg. Rank improvement	count of [ABB] Rank increase	Avg. Rank decrease	count of [ABB] Rank decrease
Wikipedia (Pre-training)	1.45	125079	0.67	-	-	-	-	-	-
Covid Dataset	1.58	16218	0.61	95.7	1.57	2	174	1.18	130
Application Review	1.42	9150	0.67	97.7	1.32	8.89	140	1.6	164
US Bill	1.51	29470	0.64	96.5	1.46	5.17	396	1.49	332
ECTHR	1.26	22295	0.67	98.5	1.25	3.24	144	1.2	254

Table 7: ABB-BERT performance on different domain data pre and post personalization phase. In pre-personalization phase, ABB-BERT was used without any domain training. The results are for short forms identified together in a sentence. m is 0.8 and Avg Diff is very close to it. The average rank without training is very high leaving little scope of big improvement. However there is improvement seen in rank of the correct option in post-personalization phase

were trained keeping $\boldsymbol{ABB\_BERT}$ parameters static for this experiment. The number of options for every $[ABB]$ was 50 for every dataset.

ABB-BERT performance results on a test set are shown in table 7. Without any training of parameter θ, ABB-BERT does very well in the pre-personalization phase. The performance on the test set in post-personalization gets better though not noticeable because the average rank is close to 1 in pre-personalization phase.

In the third experiment, we wanted to compare our work with existing work. We could not find the exact equivalent for this work, but we still decided to baseline this work for contractions using NeuSpell by Jayanthi et al. (2020). Neuspell is an excellent algorithm for misspellings, but when exposed to contractions, it makes many mistakes. Results of the baseline can be found in table 6. As expected, NeuSpell does well for lengthiest contractions than shortest contractions. The performance of Neuspell was close to 0 for all the abbreviation datasets. Hence, the results are not shown in the table 6.

For abbreviations, we tested out the algorithm on the SciAD dataset from Veyseh et al. (2020). ABB-BERT, without training, gives an average rank of 1.76, which is lower compared to the best model by Jin et al. (2019b). It is because the loss function of the algorithm is designed for ranking on large number of options. However, the performance improves after the personalization phase, with an average ranking of 1.55.

3.3 Visualizing ABB-BERT results

In the datasets, ABB-BERT is given only 50 options. It does a great job in predictions with more than 90% performance for the top 3 choices. If we look at the results, we see that most of the time, the one of top choices can make a correct substitution for $[ABB]$ in a sentence. Table 8 shows the options that scored high and make much sense. It shows that model can learn grammar and understands language well. However, the model does not consider commonsense or missing context in ranking. Table 9 shows where the model makes mistakes in predictions because of inherent challenges in this task.

4 Conclusions and future work

In this work, we propose ABB-BERT for abbreviation and contraction disambiguation. ABB-BERT tackles both of these irregularities in the text simultaneously and has the advantage that it takes into account the context in the sentence when ranking the possible alternatives. We designed it on Wikipedia and tested it on domain data also. The model may have a hard time getting the right options if they are grammatical appropriate based on context but maybe wrong by commonsense. Future work can help improve the model and suggest all *[ABB]* at the same time based on commonsense and missing context like geographical location, history

Original Sentence	With $[ABB]$	top 5 alternatives and cosine scores
Young redheaded man holding two bicycles near beach.	Young redheaded man holding [ABB] [ABB] near beach.	**ABB1**:(**two**, 0.99), (twag, 0.20), (twili,0.20), (tmfw,0.20), (townian,0.20) **ABB2**:(**bicycles**: 0.86), (berchy: 0.20), (binchy: 0.20), (bakley: 0.20), (besyde: 0.20)
This problem has been **previously** studied for zero-shot object **recognition** but there are several key differences.	This problem has been [ABB] studied for zero-shot object [ABB] but there are several key differences.	**ABB1**:(**previously**, 0.99), (provincial, 0.98), (privateering, 0.2), (pāval, 0.2), (primavera, 0.2) **ABB2**: (**recognition**, 0.99), (recréation, 0.26), (retroactive, 0.21), (rectification, 0.21), (revoluction, 0.20),
a **vivid cinematic** portrait.	a [ABB] [ABB] portrait.	**ABB1**:(**vivid**, 0.99), (vmvs, 0.20), (vhvi, 0.20), (vitruvius, 0.20), (vouvantes, 0.20) **ABB2**: (**cinematic**, 0.99), (christini, 0.20), (ciston, 0.20), (coeffient, 0.20), (clairant, 0.20)

Table 8: Selected examples of GLUE Benchmark datasets. The models made an accurate predictions on the options it was given. The model understands grammar and takes in context in the sentence

Original Sentence	With $[ABB]$	top 5 alternatives and cosine scores
"It's our judgment that the possible avenues to a peaceful resolution were **not** fully explored at the Tokyo conference," **U.S. State Department** spokesman **Richard Boucher** said.	" It's our judgment that the possible avenues to a peaceful resolution were [ABB] fully explored at the Tokyo conference," [ABB] spokesman [ABB] said.	**ABB1**: (**not** ,0.99), (nudator, 0.204), (nafat, 0.204),(ndkt, 0.204), (nonotic, 0.203) **ABB2**:(United States Delegation, 0.99), (Ukrainian Second Division, 0.99),(Ukrainian Soviet Division, 0.99), (Ukrainian Social Democratic, 0.99), (Union of Social Democrats, 0.99), **ABB3**: (road between,0.99), (Rob Bradley, 0.99), (Rosario Blanco, 0.99), (Ralph Barbara,0.99), (Roger Barclay,0.99)
Maude and Dora saw a **train** coming	[ABB] saw a [ABB] coming	**ABB1**:(Mountain Daughter,0.99), (Mo Due, 0.99), (Molino Dam, 0.99), (Mustard Digital, 0.99), (**Maude and Dora**, 0.99) **ABB2**:(**train**, 0.99), (tegne, 0.20), (tanshi, 0.20), (thunderer, 0.20), (trumain, 0.20), (tenelea, 0.20),
Alan J. Konigsberg is related to **Levy Phillips & Konigsberg**	[ABB] [ABB] related to [ABB]	**ABB1**:(All Japan Kick,0.99), (Archbishop John Kemp,0.52), (Arbab Jehangir Khan,0.35), (American John Kendrick,0.3), (Albert James Kingston,0.29) **ABB2**:(is, 0.99), (istd, 0.20), (isthmo, 0.20), (inscs, 0.20), (istres,0.20), **ABB3**:(Lord Palatine of Kyiv, 0.99), (Liverpool Park Keepers, 0.99), (La Palabra Kilometros, 0.99), (Long Pine Key, 0.99), (Lalitha Priya Kamalam,0.91)

Table 9: Selected examples of GLUE Benchmark datasets. The sentences are hard to predict because of the model outcome might be correct grammatically but does not match the ground truth. Is some cases enough information is not provided as input to make a correct prediction

of notes etc.

References

Ilias Chalkidis, Ion Androutsopoulos, and Nikolaos Aletras. 2019. Neural legal judgment prediction in English. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4317–4323, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019a. Bert: Pre-training of deep bidirectional transformers for language understanding.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019b. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic bert sentence embedding.

Giovanni Grano, Andrea Di Sorbo, Francesco Mercaldo, Corrado A. Visaggio, Gerardo Canfora, and Sebastiano Panichella. 2017. Android apps and user feedback: a dataset for software evolution and quality

improvement. In *Proceedings of the 2nd ACM SIGSOFT International Workshop on App Market Analytics*, pages 8–11.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention.

Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. 2020. Neuspell: A neural spelling correction toolkit.

Qiao Jin, Bhuwan Dhingra, William Cohen, and Xinghua Lu. 2019a. Probing biomedical embeddings from language models. In *Proceedings of the 3rd Workshop on Evaluating Vector Space Representations for NLP*, pages 82–89, Minneapolis, USA. Association for Computational Linguistics.

Qiao Jin, Jinling Liu, and Xinghua Lu. 2019b. Deep contextualized biomedical abbreviation expansion. In *BioNLP@ACL*.

Venkata Joopudi, Bharath Dandala, and Murthy Devarakonda. 2018. A convolutional route to abbreviation disambiguation in clinical text. *Journal of Biomedical Informatics*, 86:71–78.

Anastassia Kornilova and Vlad Eidelman. 2019. Billsum: A corpus for automatic summarization of us legislation.

Markus Kreuzthaler, Michel Oleynik, Alexander Avian, and Stefan Schulz. 2016. Unsupervised abbreviation detection in clinical narratives. In *Proceedings of the Clinical Natural Language Processing Workshop (ClinicalNLP)*, pages 91–98, Osaka, Japan. The COLING 2016 Organizing Committee.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Irene Z Li, Michihiro Yasunaga, Muhammed Yavuz Nuzumlali, C. Caraballo, Shiwani Mahajan, H. Krumholz, and Dragomir R. Radev. 2019. A neural topic-attention model for medical term abbreviation disambiguation. *ArXiv*, abs/1910.14076.

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

Timo Möller, Anthony Reina, Raghavan Jayakumar, and Malte Pietsch. 2020. Covid-qa: A question answering dataset for covid-19. In *Proceedings of the 1st Workshop on NLP for COVID-19 at ACL 2020*.

Chunguang Pan, Bingyan Song, Shengguang Wang, and Zhipeng Luo. 2021. Bert-based acronym disambiguation with multiple training strategies.

Fei Tan, Yifan Hu, Changwei Hu, Keqian Li, and Kevin Yen. 2020. TNT: Text normalization based pre-training of transformers for content moderation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4735–4741, Online. Association for Computational Linguistics.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Quan Hung Tran, and Thien Huu Nguyen. 2020. What does this acronym mean? introducing a new dataset for acronym identification and disambiguation. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 3285–3301. International Committee on Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018a. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Feng Wang, Weiyang Liu, Haijun Liu, and Jian Cheng. 2018b. Additive margin softmax for face verification. *CoRR*, abs/1801.05599.

Yinfei Yang, Gustavo Hernandez Abrego, Steve Yuan, Mandy Guo, Qinlan Shen, Daniel Cer, Yun hsuan Sung, Brian Strope, and Ray Kurzweil. 2019. Improving multilingual sentence embedding using bi-directional dual encoder with additive margin softmax.