

DESCRIPTION

TITLE:

METHOD AND SYSTEM FOR PERFORMING INTENT DETECTION AND CLASSIFICATION

5

Reference To Related Application

[0001] This application claims the benefit of priority of Singapore patent application
10 No. 10202400226W, filed 26 January 2024, the content of it being hereby incorporated by
reference in its entirety for all purposes.

Technical Field

15 [0002] Various embodiments relate to a method and a system for performing intent
detection and classification. Further embodiments relate to a method for training a language
model for intent detection and classification.

Background

20

[0003] Recent rapid advances in pre-trained language models, such as Bidirectional
Encoder Representation from Transformer (BERT) and Generative pre-trained transformer
(GPT), in natural language processing (NLP) have greatly improved the efficacy of text
classifiers, easily surpassing human level performance in standard datasets e.g. General
25 Language Understanding Evaluation (GLUE). However, most of these standard tasks
implicitly assume a closed-world situation, where all testing data are supposed to lie in the
same scope or distribution of the training data.

[0004] On the other hand, the issue of detecting out-of-distribution (OOD) samples has
received relatively less attention thus far. In NLP, OOD detection is the task for a
30 supervised model, trained on a particular intent dataset, to detect instances which do not
belong to the dataset used for training. In other words, out-of-distribution (OOD) detection

is the task of detecting when an input data point lies beyond the scope of the seen training set. This challenge is becoming increasingly important as NLP agents are being deployed ubiquitously in our daily lives, ranging from online chatbots, or virtual assistants to smartphone assistance app, and the potential consequences of an agent providing an answer it knows nothing about may be disastrous as it may spread disinformation and even hate speech. Hence, ensuring models accurately detect OOD data is a vital fail-safe for any task-oriented intent classifier deployed in the real world, making them more robust, and safe to use.

[0005] This has long attracted on-going efforts from the research community to build robust and accurate algorithms and models for OOD detection. Recent work may be broadly categorized into two orthogonal approaches – data generative/augmentative methods and threshold/boundary learning. For example, focuses may be on creating data that represents OOD samples to train as labelled outliers. This may be done by designing algorithms to generate samples based on the given training set, or by directly using existing, unrelated public datasets as OOD samples. By training the model on these pseudo-negative samples together with the provided positive samples, significant improvements over the state-of-the-art accuracy for OOD detection have been observed.

[0006] Earlier work on OOD detection such as Local Outlier Factor (LOF) largely relied on statistical measures, like density of the data points in an embedded space, to perform OOD detection. More recent work may be broadly categorized into the two orthogonal (and not necessarily mutually exclusive) approaches of data generative/augmentative methods and threshold/boundary learning.

[0007] Since access is only available to in-domain data, a natural question to ask is whether one can generate synthetic OOD data based on the distribution of the in-domain data. Naive – but effective – data augmentation may be easily done by treating unrelated data as outliers. For example, ImageNet may be used to augment CIFAR in computer vision, and WikiText may be used to augment AG Newsgroup in NLP, citing significant gains in outlier detection in both cases. Another existing technique operates on the latent space by sampling points on straight lines connecting in-domain samples of different classes. This in effect aims to form a convex hull filled with outlier samples, with the vertices as the known classes. The training set may be augmented with several large public datasets, which

is shown to further improve the performance. This existing technique was further tested for few-shot OOD detection, where it was augmented with latent representation generation via denoising autoencoder.

[0008] On the other hand, there were also explorations on setting the baseline by using confidence score from the outputted softmax probability to determine whether a sample is OOD, but this requires manually setting a threshold. Margin loss was introduced to maximize the decision margin in the angular space, whence making it easier for existing methods, such as LOF, to distinguish outliers. To address the issue of tuning a threshold level, subsequent work has proposed a boundary loss function to learn the radii of spherical decision boundaries around the centroid of each in-domain class. The loss is constructed to balance both the empirical risk (false positive) and the open space risk (false negative), further improving the results on OOD detection.

[0009] One of the challenges in generating OOD samples lies in the difficulty in modelling an unseen class that may be arbitrarily large and diverse, only available with a small, finite dataset. A solution to tackle this challenge may be to improve the data efficiency, by exploring prompt-based learning. This is a rapidly developing range of techniques in NLP where text prompts (i.e. essentially sentences with some blank slots) are presented to a pre-trained language model, which then proceeds to fill the slots. Work have shown that if the text prompt is formulated correctly, the language model may achieve much better zero or few-shot accuracy than fine-tuning the model in the full supervised scenario, thus achieving far higher data efficiency.

[0010] The intuition behind prompt-based methods lies in the ability to transfer the knowledge learned during pre-training, of which the data size is in orders of magnitude larger than the training data for a typical downstream task.

[0011] For example, with NLP research dominated by large pre-trained models such as BERT and its variants over the recent years, the de facto approach to addressing a downstream task, such as text classification or translation, is to fine-tune these language models based on the task data. However, while these language models may be already more data efficient than previous approaches, this still means that labelled data are required for each task to be solved.

[0012] This inconvenience paved way to the exciting paradigm of prompt-based learning. The main idea is to reformulate the task at hand to look like the original data that the language model was trained on. This usually involves generating a textual prompt and feeding it into the language model and expecting the model to output a distribution over the vocabulary rather than the classes. The vocabulary distribution is then reformatted accordingly to obtain the class prediction.

[0013] The concept of prompt-based learning first appeared at the scene with huge pre-trained language models, such as GPT-2. Other existing work have shown that these language models can perform well on standard NLP datasets such as GLUE with minimal or even no extra fine-tuning. The upshot is that by reformulating the task to something that the model has already seen during pre-training, which is consensually known as prompt engineering, one can much better leverage or transfer the knowledge within the pretraining corpora, which are of many magnitudes larger than the task data.

[0014] The biggest advantage of prompt-based learning is that a single language model trained entirely in an unsupervised manner may be used to solve a large number of tasks if given the appropriate prompts. However, the main disadvantages (as of the current knowledge) are two folds: firstly, a successful prompting model requiring good prompt engineering, which is referred by some as “an art that takes time and experience”; and secondly, while prompting methods frequently outperform fully supervised models on zero or few-shots environments, with enough labelled data fully supervised models still generally outperform prompting methods. This is especially apparently in more complicated tasks.

[0015] Another advocate of prompting methods is that pre-trained language models such as BERT are prone to catastrophic forgetting. This is the phenomenon when a language model is fine-tuned on a downstream task, the newly acquired knowledge replaces some of the linguistic knowledge previously learned during pre-training, potentially causing robustness issues in the resulting model. The use of probing approach in prompting minimizes the risks of catastrophic forgetting, especially in zero or few-shot environments. Prompt-based approaches have been applied to many tasks, such as text classification, entity recognition, information extraction, machine translation, semantic parsing, to name a few. But while these show the strength of prompting with very few samples, it is also

commonly shown that these methods often fall just short of the state-of-the-art benchmarks achieved by the full-sized training set.

[0016] In other words, one common downside that prompting presents in the existing work is the advantages over supervised learning become less obvious in high resources environments. For example, while it has been reported that high accuracies of 93.0% and 49.5% were achieved for the GLUE task SST-2 and SST-5, respectively, these figures still correspondingly represent about 40% and 20% relative drops when compared to fine-tuning with the full training set. There has been limited work so far which explores the scalability of prompt-based methods to richer datasets and whether these methods may be tuned to outperform fully supervised counterparts in these situations.

[0017] Hence, there is a need for a method and system for performing prompt-based OOD intent detection, that address at least the problems mentioned above.

Summary

[0018] According to an embodiment, a method for performing intent detection and classification is provided. The method may include receiving an input text; reformulating the input text as one or more prompts; passing the one or more prompts through a language model to obtain one or more states based on the input text, wherein the language model may include a classifier head and a language modeling head configured to operate in parallel with the classifier head; based on the one or more states, predicting a classifier output by the classifier head, and simultaneously generating a language modeling prediction by the language modeling head; and combining the classifier output and the language modeling prediction to generate an output prediction. The output prediction may be out-of-distribution if the language modeling prediction is indicative that the one or more states are predicted to be outside an in-domain label space, or otherwise, the output prediction may be an in-distribution intent. The classifier output may include the in-distribution intent falling within the in-domain label space.

[0019] According to an embodiment, a system for performing intent detection and classification using a language model including a classifier head and a language modeling head configured to operate in parallel with the classifier head is provided. The system may

include a memory; and at least one processor unit in communication with the memory. The at least one processor unit may be configured to: receive an input text, reformulate the input text as one or more prompts, pass the one or more prompts through the language model to obtain one or more states based on the input text, and combine a classifier output and a language modeling prediction to generate an output prediction. Based on the one or more states, the classifier head may be configured to predict the classifier output, and simultaneously, the language modeling head may be configured to generate the language modeling prediction. The output prediction may be out-of-distribution if the language modeling prediction is indicative that the one or more states are predicted to be outside an in-domain label space, or otherwise, the output prediction may be an in-distribution intent. The classifier output may include the in-distribution intent falling within the in-domain label space.

[0020] According to an embodiment, a method for training a language model for intent detection and classification is provided. The method may include jointly training the language model with a classification task and a language modeling task; and using at least two outlier samples. One of the at least two outlier samples may be generated by a first technique and another of the at least two outlier samples may be generated by a second technique different from the first technique.

[0021] According to an embodiment, it may be provided a non-transitory computer-readable storage medium including computer readable instructions operable when executed by a computer to perform intent detection and classification. The computer readable instructions may be configured to perform a method for performing intent detection and classification and/or a method for training a language model for intent detection and classification, according to various embodiments.

Brief Description of the Drawings

[0022] In the drawings, like reference characters generally refer to like parts throughout the different views. The drawings are not necessarily to scale, emphasis instead generally being placed upon illustrating the principles of the invention. In the following description,

various embodiments of the invention are described with reference to the following drawings, in which:

[0023] FIG. 1 shows a flow chart illustrating a method for performing intent detection and classification, according to various embodiments.

5 **[0024]** FIG. 2 shows a schematic representation of a system for performing intent detection and classification, according to various embodiments.

[0025] FIG. 3 shows a flow chart illustrating a method for training a language model for intent detection and classification, according to various embodiments.

10 **[0026]** FIG. 4 shows a flow chart illustrating an outline of the proposed method, according to one example.

[0027] FIG. 5 shows a plot illustrating the amount of training data versus the macro-F1 scores achieved by the proposed model and the various baselines (DCL, ABD, ODIST) for the CLINC dataset at the proportion of 25%, according to one example.

15 **[0028]** FIG. 6 shows a plot illustrating the amount of training data versus the macro-F1 scores achieved by the proposed model and the various baselines (DCL, ABD, ODIST) for the CLINC dataset at the proportion of 50%, according to one example.

[0029] FIG. 7 shows a plot illustrating the amount of training data versus the macro-F1 scores achieved by the proposed model and the various baselines (DCL, ABD, ODIST) for the CLINC dataset at the proportion of 75%, according to one example.

20 **[0030]** FIG. 8 shows a plot illustrating the mean and standard deviation of the various prompts used on Banking dataset 50%, according to various examples.

Detailed Description

25 **[0031]** The following detailed description refers to the accompanying drawings that show, by way of illustration, specific details, and embodiments in which the invention may be practiced. These embodiments are described in sufficient detail to enable those skilled in the art to practice the invention. Other embodiments may be utilized and structural, logical, and electrical changes may be made without departing from the scope of the invention. The
30 various embodiments are not necessarily mutually exclusive, as some embodiments can be combined with one or more other embodiments to form new embodiments.

[0032] Embodiments described in the context of one of the methods or devices are analogously valid for the other methods or devices. Similarly, embodiments described in the context of a method are analogously valid for a device, and vice versa.

[0033] Features that are described in the context of an embodiment may correspondingly be applicable to the same or similar features in the other embodiments. Features that are described in the context of an embodiment may correspondingly be applicable to the other embodiments, even if not explicitly described in these other embodiments. Furthermore, additions and/or combinations and/or alternatives as described for a feature in the context of an embodiment may correspondingly be applicable to the same or similar feature in the other embodiments.

[0034] In the context of various embodiments, the articles “a”, “an” and “the” as used with regard to a feature or element include a reference to one or more of the features or elements.

[0035] In the context of various embodiments, the term “about” or “approximately” as applied to a numeric value encompasses the exact value and a reasonable variance.

[0036] As used herein, the term “and/or” includes any and all combinations of one or more of the associated listed items.

[0037] As used herein, the phrase of the form of “at least one of A or B” may include A or B or both A and B. Correspondingly, the phrase of the form of “at least one of A or B or C”, or including further listed items, may include any and all combinations of one or more of the associated listed items.

[0038] As used herein, the expression “configured to” may mean “constructed to” or “arranged to”.

[0039] Various embodiments may provide a model based on prompt-based learning that is able to leverage the advantages of strong discriminativeness from traditional supervised learning, as well as few-shot capabilities from prompting methods. While prompt-based learning typically struggles to fully utilize a large dataset for training, this issue may be addressed via supplementing it with supervised learning.

[0040] More precisely, various embodiments provide the training of a language model to perform classification as well as OOD detection simultaneously with two separate output heads. Essentially, a traditional classifier head may be used for classification as in fully supervised learning, whereas a prompting head may be utilized to perform outlier detection.

[0041] FIG. 1 shows a flow chart illustrating a method 100 for performing intent detection and classification, according to various embodiments.

[0042] As seen in FIG. 1, at Step 102, an input text may be received. At Step 104, the input text may be reformulated as one or more prompts. At Step 106, the one or more prompts
5 may be passed through a language model to obtain one or more states based on the input text. The language model may include a classifier head and a language modeling head configured to operate in parallel with the classifier head. The language modeling head may be different (or separate) from the classifier head.

[0043] The one or more states may be passed through the classifier head and the language
10 modeling head simultaneously. At Step 108, based on the one or more states, a classifier output may be predicted by the classifier head, and a language modeling prediction may be simultaneously generated by the language modeling head. At Step 110, the classifier output and the language modeling prediction may be combined to generate an output prediction. The output prediction may be out-of-distribution if the language modeling prediction is
15 indicative that the one or more states are predicted to be outside an in-domain label space, thereby performing OOD intent detection, or otherwise, the output prediction may be an in-distribution intent. The classifier output may include the in-distribution intent falling within the in-domain label space, thereby performing intent classification.

[0044] In the context of various embodiments, the in-domain label space may include a
20 label space formed by performing traditional intent classification, e.g. by the classifier head, to classify the input text to one or more corresponding labels.

[0045] The output prediction may also be referred to as an intent class.

[0046] In other words, given an intent text classification dataset $\{x_i, y_i\}$, where $x_i \in \mathcal{X}$ is a plain input text and $y_i \in \mathcal{Y}'$ is its corresponding label, a traditional intent classification task
25 aims to classify inputs $x \in \mathcal{X}$ into one of the labels in $\mathcal{Y}'$. The task of OOD detection (or interchangeably referred to as OOD intent detection) takes this one step further by considering inputs x whose label does not necessarily lie in $\mathcal{Y}'$. If the label set is extended with a special tag “ood” and denote the extended label space $\mathcal{Y} = \mathcal{Y}' \cup \{\text{ood}\}$, then OOD detection is the classification task to map inputs from $\mathcal{X}$ into a class from $\mathcal{Y}$.

[0047] In the context of various embodiments and examples, the term “OOD” (or “ood”) denoting “out-of-distribution” may interchangeably be referred to as “OOS” (or “oos”) denoting “out-of-scope”.

[0048] In various embodiments, each of the one or more prompts may include an input slot and an answer slot. Reformulating the input text as the one or more prompts at Step 104 may include filling or replacing the input slot with the input text. The method 100 may further include mapping the output prediction to a set of tokens (e.g. via a verbalizer); determining a token with a highest average (uniform) probability or a majority voting from the set of tokens; and filling the answer slot with the determined token. In other examples, the step of determining the token from the set of tokens may be performed using weighted averaging or knowledge distillation.

[0049] A prompt (i.e. a text prompt or template) is a textual string with two (or more) slots. These are usually referred to as the input and answer slots, and may be denoted by [X] and [Z], respectively. Given an input sequence $x \in \mathcal{X}$, the slot [X] is filled, or replaced, by x before passing through a pre-trained language model. The aim of the model is to fill the answer slot [Z] with the most appropriate choice from a pre-defined answer set Z . The set Z is usually constructed via a verbalizer, which maps each class in $\mathcal{Y}$ to a set of words or tokens (e.g. the “positive” class may be mapped to the set {“good”, “great”, “awesome”, “incredible”, ...} in sentiment analysis). The outputted answer $z \in Z$ is eventually mapped onto the label (answer) space $\mathcal{Y}$ of the original task. This eventual step may be redundant for tasks such as text classification, since Z is often simply the set of labels in the label space $\mathcal{Y}$. The prompt and answer designs are highly dependent on the task at hand and their constructions may often be referred as prompt and answer engineering, respectively.

[0050] In various embodiments, the one or more prompts may include at least one of the following: a first comment about the input text and a first question concerning the token; a second comment about the input text and a second question concerning the token, the second comment being different from the first comment and the second question being different from the first question; a third comment about the input text and a topic concerning the token, the third comment being the same or different from the first comment or the second comment or both the first comment and the second comment; a first query about the input text and a user enquiry about the token; or a second query about the input

text and an enquiry about the token, the first query being different from the second query and the user enquiry being different from the enquiry. It should be appreciated that in the event where more than one prompt is involved, these prompts may be passed through the language model in no particular order to obtain the one or more states. More specific
5 examples of the prompt(s) will be described later.

[0051] Reformulating the input text as one or more prompts at Step 104 may further include pre-processing the input text. For example, pre-processing the input text may include removing a non-alphanumeric character (e.g. punctuation) and/or a stopword originating from the input text.

10 **[0052]** In various embodiments, the language model may further include a pre-trained model in upstream communication with the classifier head and the language modeling head, and the one or more states may include one or more hidden states generated by the pre-trained module. Examples of the pre-trained model may include a Bidirectional Encoder Representation from Transformer (BERT), or a Generative pre-trained
15 transformer (GPT), or a Pathways Language Model (PaLM), or a Large Language Model Meta AI (Llama), or Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA), or Generalized Autoregressive Pretraining for Language Understanding (XLNet), a Robustly Optimized BERT Pretraining Approach (RoBERTa), or a Decoding-enhanced BERT with Disentangled Attention (DeBERTa).

20 **[0053]** The language model may be jointly trained with a classification task and a language modeling task and using at least two outlier samples. One of the at least two outlier samples may be generated by a first technique and another of the at least two outlier samples may be generated by a second technique different from the first technique.

[0054] Essentially, two distinct types of outliers may be generated for training.

25 **[0055]** For example, the first technique may include performing a convex combination of a first embedding (h_x) associated with a first training input (x) and a second embedding ($h_{x'}$) associated with a second training input (x'), the first training input and the second training input being from different classes, and wherein an embedding (h) of the outlier sample generated by the first technique is represented by: $h = \theta h_x + (1-\theta) h_{x'}$, where a
30 distribution of θ is a uniform distribution given by $\theta \sim U(0, 1)$ that may be sampled uniformly.

[0056] In other words, the first technique may take convex combinations of embeddings with different classes. It can be shown that this generates “harder” outliers which improve the discriminativeness of the classifier and may be adopted to the prompting dimension. Given the two inputs x and x' from different classes, the embeddings h_x and $h_{x'}$ may be extracted from the token in the answer slot $[Z]$ to construct the outlier embedding h . The outlier embedding h may then be fed into the language modeling head with Z_{oos} (out-of-scope answer) as the target. Z_{oos} may be interchangeably referred to as Z_{ood} (out-of-distribution answer).

[0057] The second technique may include computing a mean value (μ) and a standard deviation (σ) of all sentence lengths in a training set of the in-domain label space; removing stopwords from the training set to construct a set of words (W); and uniformly sampling from a uniform distribution of the set of words (W) to form the outlier sample including a generic sample represented by: $x_i \sim U(W)$ for $0 < i \leq l$, where $l \sim \mathcal{N}(\mu, \sigma^2)$. $\mathcal{N}(\mu, \sigma^2)$ is the normal distribution with μ and variance σ^2 . For example, the training set may include the first training input (x).

[0058] Basically, the second technique may be for generating inputs to represent the “generic” sample of that domain, i.e. $x = (x_1, \dots, x_l)$.

[0059] Several sampling strategies, including weighted sampling based on word frequencies, have been evaluated. However, no significant differences were found in the results. In practice, l may be limited to $5 \leq l \leq M$, where M is the maximum length of the corpus.

[0060] Some examples are shown in Table I depicting the dataset statistics and examples of answer set Z_c and generic samples.

[0061] Table I

Dataset	Classes	Size	Z_c Sample	Generic Sample
Banking	77	13,083	{card, payment, fee, charged}	“cancelled apple verify need refund...”
OOS/CLINC	150	23,700	{change, volume}	“take start want insurance need couple pizza...”
StackOverflow	20	20,000	{osx}	“studio pivot scala dispose compiling soap...”

[0062] The purpose of these outliers are two-folds: firstly, the outliers help augment the coverage of OOD samples in the cases where convex combinations are insufficient, especially in situations with fewer seen classes. Secondly, the outliers help to unbias the probability of predicting certain words which are ubiquitous in that domain.

[0063] For example, the Banking dataset contains various enquiries into banking related questions, whence unseen samples may be inherently biased towards one of the inlier classes by containing generic terms such as “money” or “transfer”. The effectiveness of these samples in the Results section below.

5 **[0064]** In various embodiments, the classification task may include applying, by the classifier head, a softmax function on a plurality of hidden states to obtain classifier logits; and training the classifier head with cross entropy loss based on the classifier logits. The plurality of hidden states may be generated by passing training prompts reformulated from a training input text of a specific class through the language model.

10 **[0065]** The language modeling task may include applying, by the language modeling head, a conditional softmax function on the plurality of hidden states to obtain language modeling logits; and training the language modeling head with binary cross entropy loss based on the language modeling logits.

[0066] In various embodiments, the at least two outlier samples may be trained with binary
15 cross entropy loss.

[0067] The language model in entirety, upon training, may have a weighted loss function including a summation of the cross entropy loss based on the classifier logits, a weighted binary cross entropy loss based on the language modeling logits, and a weighted binary cross entropy loss associated with the at least two outlier samples.

20 **[0068]** In various embodiments, the language model may be post-trained with an adaptive decision boundary. In other words, the method 100 may include post-training the language model with an adaptive decision boundary.

[0069] While the method described above is illustrated and described as a series of steps or events, it will be appreciated that any ordering of such steps or events are not to be
25 interpreted in a limiting sense. For example, some steps may occur in different orders and/or concurrently with other steps or events apart from those illustrated and/or described herein. In addition, not all illustrated steps may be required to implement one or more aspects or embodiments described herein. Also, one or more of the steps depicted herein may be carried out in one or more separate acts and/or phases.

30 **[0070]** FIG. 2 shows a schematic representation of a system 220 for performing intent detection and classification, according to various embodiments.

[0071] As shown in FIG. 2, the system 220 includes a memory 222; and at least one processor unit 224 in communication with the memory 222, as denoted by a line 226. The at least one processor unit 224 may be configured to: receive an input text, reformulate the input text as one or more prompts, pass the one or more prompts through a language model to obtain one or more states based on the input text, and combine a classifier output and a language modeling prediction to generate an output prediction. The language model may include a classifier head and a language modeling head configured to operate in parallel with the classifier head. The at least one processor unit 224 may be configured to pass the one or more states through the classifier head and the language modeling head simultaneously. The output prediction may be out-of-distribution if the language modeling prediction is indicative that the one or more states are predicted to be outside an in-domain label space, or otherwise, the output prediction may be an in-distribution intent, wherein the classifier output may include the in-distribution intent falling within the in-domain label space. When the output prediction is out-of-distribution, the system 220 may perform OOD intent detection. When the output prediction is an in-distribution intent, the system 220 may perform intent classification, the in-domain label space being a label space formed by performing traditional intent classification, e.g. by the classifier head, to classify the input text to one or more corresponding labels. Each of the one or more prompts may include an input slot for filling the input text; and an answer slot.

[0072] The system 220 may include the same or like elements or components as those of the method 100 of FIG. 1, and as such, the like elements may be as described in the context of the method 100 of FIG. 1, and therefore some corresponding descriptions are omitted here.

[0073] The system 220 may further include a verbalizer configured to map the output prediction to a set of tokens. The at least one processor unit 224 may be further configured to determine a token with a highest average (uniform) probability or a majority voting from the set of tokens and fill the answer slot with the determined token. In other examples, the at least one processor unit 224 may be further configured to determine the token with weighted averaging or knowledge distillation from the set of tokens and fill the answer slot with the determined token.

[0074] The at least one processor unit 224 may be further configured to remove a non-alphanumeric character (e.g. punctuation) and/or a stopword originating from the input text.

[0075] In various embodiments, the language model may further include a pre-trained model in upstream communication with the classifier head and the language modeling head, the pre-trained model being configured to generate the one or more states. the pre-trained model comprises a BERT, or a GPT, or a PaLM, or a Llama, or ELECTRA, or XLNet, a RoBERTa, or a DeBERTa. The language model may be jointly trained with a classification task and a language modeling task, and using at least two outlier samples.

[0076] During training with the classification task, the classifier head may be configured to apply a softmax function on a plurality of hidden states to obtain classifier logits, and configured to be trained with cross entropy loss based on the classifier logits. The plurality of hidden states may be generated by passing training prompts reformulated from a training input text of a specific class through the language model.

[0077] During training with the language modeling task, the language modeling head may be configured to apply a conditional softmax function on the plurality of hidden states to obtain language modeling logits, and configured to be trained with binary cross entropy loss based on the language modeling logits.

[0078] The language model may be configured to be post-trained with an adaptive decision boundary.

[0079] The at least two outlier samples may be configured to be trained with binary cross entropy loss.

[0080] FIG. 3 shows a flow chart illustrating a method 340 for training a language model for intent detection and classification, according to various embodiments.

[0081] As seen in FIG. 3, at Step 342, the language model may be jointly trained with a classification task and a language modeling task. At Step 344, at least two outlier samples may be used. One of the at least two outlier samples may be generated by a first technique and another of the at least two outlier samples may be generated by a second technique different from the first technique.

[0082] In various embodiments, using the at least two outlier samples at Step 344 may include training the at least two outlier samples with binary cross entropy loss.

[0083] The method 340 may include the same or like elements or components as those of the method 100 of FIG. 1, and as such, the like elements may be as described in the context of the method 100 of FIG. 1, and therefore the corresponding descriptions are omitted here.

[0084] For example, the first technique, the second technique, the classification task, the language modeling task of the method 340 of FIG. 3 are described in similar context to the first technique, the second technique, the classification task, the language modeling task of the method 100 of FIG. 1, respectively.

[0085] Various embodiments may provide a non-transitory computer-readable storage medium including computer readable instructions operable when executed by a computer to perform intent detection and classification, the computer readable instructions configured to perform the method 100 of FIG. 1 and/or the method 340 of FIG. 3.

[0086] Examples and studies of the method and system for performing intent detection and classification as well as the method for training a language model for intent detection and classification will be described below.

[0087] OOD detection is vital for any NLP applications deployed in the real world. Various embodiments, examples and model described herein provides a method for OOD detection based on prompting, which is known for its zero and few-shot capabilities. To the best of the inventors' knowledge, this is the first work to employ prompting methods for OOD detection. This is also a non-trivial derivation as a naive approach of combining prompt-based learning and supervised learning would yield suboptimal performance that is at best on-par with existing works.

[0088] By training the model jointly with classification and language modeling tasks, as well as using two different approaches to generate outlier samples, this method surpasses state-of-the-art results on the vast majority of the settings in the tested datasets.

[0089] Generating synthetic outliers in terms of prompts allows the model to more efficiently learn OOD samples than the existing methods. The two methods, that have been used to generate OOD samples, involves reformulating to operate on the prompting level, and sampling based on the word distribution of the entire training corpus, that serves to represent generic sentences in the same domain as the training corpus.

[0090] Further analysis on varying the training set shows that the model is indeed able to capitalize from both the benefits of prompting and supervised learning, resulting in a model

that is far more consistent on both low and high resource environments – an uncommon phenomenon in prompt-based methods thus far. For example, comparison has been made with other prompting methods used by large language models such as GPT3.5, and the results of are found to be overall competitive, while only consuming a negligible fraction of the resources required. Experiments have also shown that the method for OOD detection based on prompting is consistently applicable from low-resources environment to scaling to full training datasets, in contrast to existing prompting approaches that often focus on low-resources challenges.

[0091] Based on extensive testing on nine different settings across three standard datasets used for OOD detection, the method for OOD detection based on prompting when augmented with a decision learning algorithm (e.g. with adaptive decision boundary) is able to achieve competitive or superior performances compared with the current state-of-the-art in all cases.

[0092] An analysis on how effective prompting is, especially in relation to fully-supervised fine-tuning, has been performed. The effectiveness of the data generation method and how the nature of the dataset affects the efficacy have been analyzed.

[0093] The extensive analysis on each dataset as well as comprehensive ablation studies on each component of the model have been performed to benefit future studies on the fast expanding field of prompt-based learning. There are several possibilities for future work. For example, while different sampling strategies were tested as described below, it is plausible that using a neural generative language model to generate generic in-domain samples may yield further improvements. This is mostly due to the cohesiveness of such generated samples (in the human perspective at least) compared with statistical sampling, which is more likely to resemble the distribution of outliers used in testing. Further, advances in training a model to perform learnable prompt and answer engineering may be explored to the task of OOD detection.

Overview and Motivation

[0094] FIG. 4 shows a flow chart illustrating an outline (or overall flow) of the proposed method 400, according to one example.

[0095] The proposed method 400 may be described in similar context to the method for performing intent detection and classification 100 of FIG. 1 and the method for training a language model for intent detection and classification 340 of FIG. 3.

[0096] As seen in FIG. 4, a pre-trained language model 401 (e.g. BERT) is fitted with two heads, namely a feedforward classifier head 403 for classification and a feedforward language modeling head 405 for prompting. During training, OOD samples are synthesized with two approaches namely, convex combinations and generic sampling 440, and both heads 403, 405 are trained jointly. During inference, the classifier result (or classification result) 408a is taken unless the feedforward language modeling (prompting) head 405 predicts that the input 402 is OOD (based on the OOD result 408b).

[0097] An input x 402 (e.g. an input text being received at Step 102 of FIG. 1) may be first reformulated as prompts (based on prompt reformulation 404) and passed through a pre-trained language model 401, outputting the hidden states (e.g. as Steps 104 and 106 of FIG. 1). The same states are passed through the two heads – one with a classifier head 403 to predict a class $y \in \mathcal{Y}'$ and the language modeling (prompting) head 405 which predicts if it is OOD (e.g. as Step 108 of FIG. 1). The two results are combined to obtain the final result 410 (e.g. as Step 110 of FIG. 1). During training, two types of synthetic outliers are generated using x 402 (e.g. as Step 344 of FIG. 3), and the entire model that includes the pre-trained language model 401, the feedforward classifier head 403, and the feedforward language modeling head 405, is trained end-to-end with a carefully constructed loss function (e.g. as Step 342 of FIG. 3).

[0098] The inspiration of the model comes from two aspects. Firstly, with sufficient data, fully-supervised models often outperform prompting methods, as discussed above. Secondly, however, generating synthetic outliers on a fully-supervised model is data inefficient, and often requires specific tuning for each dataset. Comparisons have been made for the second aspect and will be discussed in more details later in the Experiments section below, where it is shown that a generative method (e.g. 400) may show stronger performance on a CLINC dataset, but its performance on a Banking dataset and a StackOverflow dataset is relatively lacking compared with the results of deep open intent classification with adaptive decision boundary.

[0099] It may be assumed that this discrepancy may be explained by the difference in the number of intents on each dataset, where having a large number of classes and training samples is required for the generative method to perform optimally. On the contrary, on datasets with fewer classes, there are not enough data to generate a comprehensive representation of OOD samples, resulting in a slightly weaker performance in comparison.

[0100] This is where prompting strives. By reformulating the classification task as a prompt 404, the pre-trained model 401 leverages the information learned previously. The benefits come in two folds. The first is that generating OOD samples based on the given training set can only cover limited scope, regardless of the method. While one could augment with other public datasets, the coverage is still nowhere as broad as the coverage of the pre-training corpus (i.e. the entire Wikipedia and common crawl). A by-product is that this method 400 may be more easily adapted to other languages without large amount of public datasets. The second is that the capability of prompting in low resource learning means that prompting is less sensitive to the ratios of generated outliers and thus is more consistent across the datasets.

Prompt Reformation

[0101] The prompt reformation 404 of FIG. 4 may be described in similar context with the input text being reformulated as one or more prompts at Step 104 of FIG. 1.

[0102] The prompt design may be detailed below.

[0103] Given an input x 402, an ensemble of prompts may be as follows:

P_1 : [X]. This question is about [Z].

P_2 : [X]. This topic is about [Z].

P_3 : [X]. This query is about [Z].

P_4 : The following is a query: [X]. The user is asking about [Z].

P_5 : The following is a query: [X]. The enquiry is about [Z].

[0104] [X] is the input slot, filled by x , and [Z] is the answer slot outputted by the model. Using the five predictions z_i 's, both majority voting as well as averaging the probabilities are tested, but insignificant differences are found. It should be appreciated that the ensemble of prompts P_1 to P_5 may be listed in any order, and not limited to the order shown in the list above.

[0105] For each intent class $y \in \mathcal{Y}$, all the words (expanding the tokenizer’s vocabulary if necessary) are taken in the description tag of the intent class as the permissible answer set, i.e.

$$\mathcal{Z}_y = \{\text{words} \in \text{tag}(y)\}.$$

5 **[0106]** It is noted that punctuations such as underscores, as well as stopwords (using natural language toolkit, NLTK), are removed as part of preprocessing. The OOD answer set is manually set as $\mathcal{Z}_{\text{ood}} = \{\text{“ood”}, \text{“other”}, \text{“unknown”}\}.$

[0107] Then the answer set $\mathcal{Z}$ is simply the union of all the tags, i.e.

$$\mathcal{Z} = \bigcup_{y \in \mathcal{Y}} \mathcal{Z}_y$$

10

[0108] Examples of $\mathcal{Z}_y$ are shown in Table I above.

[0109] Given an output $z(\cdot)$ from the language model (a probability distribution over $\mathcal{Z}$ rather than the entire vocabulary), the prediction, denoted by $\hat{z}$, of the instance is given by the class y with the highest average probability over all the words in $\mathcal{Z}_y$, i.e.

15

$$\hat{z} = \max_{y \in \mathcal{Y}} \left(\frac{1}{|\mathcal{Z}_y|} \sum_{w \in \mathcal{Z}_y} z(w) \right)$$

Joint Training

[0110] The entire model is trained as follows. Given an input x 402 from class y , it is first reformulated as prompts and passed through a language model (LM) as

$$h = \text{LM}(x)$$

20 to obtain the hidden states. Then, a classifier feedforward layer 407 is added to obtain the logits

$$\mathcal{Y} = \text{softmax}(\text{FF}_{\text{cls}}(h))$$

and trained with cross entropy loss

$$\mathcal{L}_{\text{ce}}(x) = -\log(\mathcal{Y}_y),$$

25 where $\mathcal{Y}_y$ is logits of class y .

[0111] The hidden states h is reused and fitted with the language modeling head 405

$$z = \text{softmax}_{w \in \mathcal{Z}}(\text{FF}_{\text{LM}}(h))$$

where the softmax 409 is taken only with respect to the indicies in the answer set $\mathcal{Z}$. This is trained with binary cross entropy loss

$$\mathcal{L}_{\text{bce}}(x) = - \sum_{w \in \mathcal{Z}_c} \log(z(w))$$

5 **[0112]** The softmax functions (e.g. at 407, 409) are current de facto choices for normalized probability distributions.

[0113] The generated outliers x_{oos} are also trained with binary cross entropy (with its target being $\mathcal{Z}_{\text{oos}}$), yielding the final weighted loss function

$$\mathcal{L}(x) = \mathcal{L}_{\text{ce}}(x) + \alpha_1 \mathcal{L}_{\text{bce}}(x) + \alpha_2 \mathcal{L}_{\text{bce}}(x_{\text{oos}})$$

where α_i are weights used to balance the various loss functions.

10 **[0114]** The joint training may be described in similar context to Step 344 of FIG. 3.

Inference

[0115] At inference, the classification predictions y and the language modeling predictions z are first obtained. Only the language modeling predictions z are used to determine
15 whether the input 402 is OOD. Otherwise, the prediction y given by the classifier head 403 is used. In other words, the final prediction $\hat{y}$ 410 is given by the following:

$$\hat{y} = \begin{cases} \text{oos} & \text{if } \text{tag}(\hat{z}) = \text{oos (at prompt ensemble 411, e.g. majority vote),} \\ \text{argmax}(y) & \text{otherwise.} \end{cases}$$

Variant with ADB

20 **[0116]** To allow for a fairer and more comprehensive comparison with state-of-the-art results, a variant of the proposed model, in which an adaptive decision boundary (ADB) is post-trained on the proposed model, is tested. The proposed model (as described herein) may be the entire model that includes the pre-trained language model 401, the feedforward classifier head 403, and the feedforward language modeling head 405 (FIG. 4). This is an
25 extra layer that may be trained on top of the proposed model and is also used in some of the other baselines (c.f. those presented in the Experiments section below). Given a model already fine-tuned on the training data (e.g. 412 of FIG. 4), this first compute class centroids c_i for $i \in \mathcal{Y}$, and then learns boundaries Δ_i for each class via the boundary loss function

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left[\delta_i (d(z'_i - c_i) - \Delta_{y_i}) + (1 - \delta_i) (\Delta_{y_i} - d(z'_i - c_i)) \right]$$

where z' denotes the output of the language model (before the prompting and classifier heads) (e.g. after 401 of FIG. 4), $d(\cdot)$ is a norm function (e.g. Euclidean or l2-norm),

$$\delta_i = \begin{cases} 1 & \text{if } d(z'_i - c_i) > \Delta_{y_i}, \\ 0 & \text{if } d(z'_i - c_i) \leq \Delta_{y_i} \end{cases}$$

5 and N is the number of samples in the training set of a particular intent class. For example, if a training data has 5 known intents classes, with 10, 20, 30, 40 and 50 samples respectively, then the value of N is correspondingly 10, 20, 30, 40 and 50 for each of the intent classes.

[0117] The prediction of an input z is either the class k with the closest centroid, or OOD
 10 if the distance to the centroid $d(z, c_k)$ is larger than Δ_k . This translates to the final prediction
 410 (FIG. 4) as

$$\hat{y} = \begin{cases} \text{ood} & \text{if } \text{tag}(\hat{z}) = \text{ood (at prompt ensemble 411),} \\ & \text{or } d(z'_i - c_i) > \Delta_i \text{ for all } i \in C, \\ \text{argmax}(y) & \text{otherwise} \end{cases}$$

where C represents the set of of all known intent classes. For example, if there are only two
 known intents “book_ticket” & “check_flight_status”, then $C = \{\text{“book_ticket”},$
 15 “check_flight_status”}.

A. Experiments

A-I. Dataset

20 [0118] The proposed model is tested on three widely used public datasets:

- **Banking** is a dataset with 77 intents and 13,083 queries on the banking domain;
- **OOS/CLINC** is a dataset with 150 intents, 22,500 in-domain queries and 1,200 OOD queries designed for OOD detection;
- **StackOverflow** is a dataset originated from Kaggle, which has 20 intents and
 25 20,000 queries from the website StackOverflow.

[0119] The original train, validation and test splits for each of the datasers can be found at
<https://github.com/thuiar/TEXTTOIR>.

[0120] The proposed model may be the entire model that includes the pre-trained language model 401, the feedforward classifier head 403, and the feedforward lanaguage modeling head 405 (FIG. 4).

5 A-II. Baselines

[0121] The experimental results are compared with the following baselines:

- **DCL** trains a $(k + 1)$ -classifier by generating convex combinations of intents. They also augment the training set with from public dataset such as Multi-Genre Natural Language Inference, MNLI.
- 10 • **ADB** crafts a boundary loss function to train spherical shaped boundaries for each inlier class. A sample is classified as OOD if it does not fall inside any of the learned spheres.
- **ODIST** generates outliers using distributionally shifting. Inputs are first chunked, with each chunk masked and replaced by the prediction of a
15 generative language model. Generated outputs which are considered contradictory to the original by the same language model are retained as OOD samples. On top of that, ADB is employed as the decision boundary.
- **In-Context Learning (ICL)** with large language models (LLMs) augments the test sentence with labelled examples, as a substitute for the high resources (or
20 inaccessibility) needed to fine-tune LLMs using traditional few-shot learning methods. In-context learning has proven to be highly efficient and is one of the de facto approaches for performing inference with LLMs.
- **Chains-of-Thoughts (CoT)** is a prompting technique originally designed to solve mathematics or logical problems by instructing the LLM to be more
25 verbose and write out the thought process before arriving at an answer.

[0122] The first three baselines use BERT as their language model, whereas ODIST utilizes BART in addition for the distributional shifting.

[0123] For LLM-based prompting approaches, the model *gpt-35-turbo-16k* provided by OpenAI for both ICL and CoT, accessed via Microsoft Azure. The reason behind this
30 choice (or rather, the lack of other alternatives) will be explained below.

A-III. Setup

[0124] Following prior practices, for each dataset, known intents are randomly sampled thrice, with proportions of 25%, 50% and 75%, respectively. Intents not sampled are grouped together (as well as the original “oos” class in CLINC) as OOD, with their training and validation samples removed.

[0125] BERT (bert-base-uncased, <https://huggingface.co/bert-base-uncased>) as the proposed language model (e.g. 401 of FIG. 4) to allow for the fairest comparison. Based on empirical results, for each batch, the two types of outliers are sampled by the ratio 1:3:3, i.e. three times the size of inliers and train until the model converges. The maximum length M of generic samples is set to 79, 28 and 41 for the Banking, CLINC and StackOverflow datasets, respectively. Learning rate is set to about $2e^{-5}$, with warm-up for first 10% of steps and decay linearly afterwards. The loss weights α_i are both set to 1 for these datasets after empirical testing. All experiments were done with the PyTorch framework.

[0126] The implementations, as well as hyperparameters and configurations whenever applicable, of ADB are adapted from the official repository (<https://github.com/thuiar/Adaptive-Decision-Boundary>).

[0127] For few-shot ICL and CoT, the model is firstly provided with a list of seen intents, followed by pairs of examples in the format of “Q: {text}\n A: {label}\n”, all as part of the system prompt. The number of shots refers to the number of examples provided per seen intent class. Furthermore, for chains-of-thoughts, up to 3 examples per class are manually annotated, and these are inserted in place of the raw label. For example, for the sentence from the Banking dataset “Why am I being charged for transfers? Are they not free?” the annotation used was “The user was surprised about being charged for transferring money, as the user expected it to be free, so the intent is transfer fee charged”. In the user prompt, a similar phrase to the simple prompt “Let’s think step by step” may be used as the instruction. Sentences which were filtered by Azure’s safety content were excluded in the computation of the results.

A-IV. Practical Choices and Considerations of LLMs

[0128] Since the research on new prompting techniques, as well as the release of more updated LLMs are occurring rapidly, the reasons for the choices behind the proposed model

and prompting technique choices are based on a practical point of view. Firstly, it should be noted that BERT models (or rather, most encoder-based pre-trained language models) are trained with the objective of masked-language modelling, and as a result are unsuitable to be prompted with techniques such as in-context learning or chains-of-thought.

5 **[0129]** As such, publicly available LLMs are explored, where OpenAI’s GPT3.5 series may be the most sensible choice as a balance between cost and accuracy. GPT4, while in theory is better in performance, is far too costly to be practical for intent detection at the moment (e.g. GPT4-32k is currently about 20 times more expensive than GPT3.5-16k).

[0130] Since in-context learning and its variants (including chain-of-thoughts, self-consistency and so on) all involve appending sufficient examples per class before the user prompt, this renders these approaches highly unsalable with the number of intents. Taking the Banking dataset with 77 intents and 3,080 testing utterances as an example, a prompt for zero-shot learning averages to around 380 tokens per utterance, and running the entire test set may cost 3.5 USD in total (<https://azure.microsoft.com/en-us/pricing/details/cognitiveservices/openai-service/>), disregarding the completion token cost as negligible in comparison. However, performing 1-shot chain-of-thought dramatically increases these numbers to 3,200 tokens per utterance and 30.2 USD respectively; and 3-shot triples these figures.

[0131] These numbers have several practical implications: firstly, while there are other open-sourced LLMs such as Llama2, these usually have a lower context length of 4,096, which is insufficient for the required purposes. Secondly, they are limited to incontext learning and chain-of-thought as comparisons. While self-consistency chain-of-thought may have proven to yield significant improvements over simple chain-of-thought, following the original implementation of taking majority vote over 40 samples may cost over 3k USD for a single run with 3-shots. Finally, these associated costs also discourage the tests with LLM to be repeated to provide a more accurate average.

B. Experimental Results and Analysis

30 **[0132]** Tables II and III show the experimental results of the proposed model, where each entry is the averages of ten runs with standard deviations.

[0133] Table II depicts the accuracy and macro F1-scores for Banking, CLINC and StackOverflow datasets with three different proportions of 25%, 50% and 75%.

[0134] Table II

	Method	Banking		OOS/CLINC		StackOverflow	
		Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
25%	DCL	74.11	69.93	88.44	80.73	68.74	65.64
	ADB	78.85	71.62	87.59	77.19	86.72	80.83
	ODIST	81.69 $\pm$ 1.43	N/A	89.79 $\pm$ 1.99	N/A	91.53 $\pm$ 1.96	N/A
	Ours w/o CC	24.77 $\pm$ 0.21	42.80 $\pm$ 1.32	33.27 $\pm$ 4.76	44.35 $\pm$ 1.95	28.32 $\pm$ 1.13	41.85 $\pm$ 4.01
	Ours w/o GS	77.44 $\pm$ 4.85	71.27 $\pm$ 3.95	90.78 $\pm$ 1.38	82.83 $\pm$ 1.75	62.68 $\pm$ 6.25	63.27 $\pm$ 3.16
	Ours w/o CLS	75.00 $\pm$ 1.24	70.54 $\pm$ 1.27	86.96 $\pm$ 3.25	78.51 $\pm$ 2.79	56.95 $\pm$ 4.85	57.76 $\pm$ 3.52
	Ours	82.00 $\pm$ 2.89	75.45 $\pm$ 2.70	91.30 $\pm$ 1.15	83.20 $\pm$ 1.46	75.78 $\pm$ 4.57	70.24 $\pm$ 4.57
	Ours w. ADB	85.79 $\pm$ 2.76	77.75 $\pm$ 2.35	92.05 $\pm$ 0.95	83.59 $\pm$ 1.84	87.78 $\pm$ 5.22	82.66 $\pm$ 5.96
50%	DCL	72.69	79.21	88.33	86.67	75.08	78.55
	ADB	78.86	80.90	86.54	85.05	86.40	85.83
	ODIST	80.90 $\pm$ 1.15	N/A	88.61 $\pm$ 0.82	N/A	88.52 $\pm$ 1.26	N/A
	Ours w/o CC	44.32 $\pm$ 3.36	61.22 $\pm$ 6.21	45.00 $\pm$ 2.71	61.68 $\pm$ 1.17	50.68 $\pm$ 0.44	62.59 $\pm$ 1.11
	Ours w/o GS	80.51 $\pm$ 1.74	83.03 $\pm$ 1.94	87.45 $\pm$ 1.83	86.62 $\pm$ 1.75	75.41 $\pm$ 9.48	79.59 $\pm$ 5.95
	Ours w/o CLS	75.21 $\pm$ 2.18	78.91 $\pm$ 0.83	84.00 $\pm$ 0.97	81.16 $\pm$ 0.60	67.34 $\pm$ 1.81	72.57 $\pm$ 0.86
	Ours	82.18 $\pm$ 1.71	83.36 $\pm$ 1.30	89.12 $\pm$ 0.89	87.27 $\pm$ 0.77	83.53 $\pm$ 2.18	83.89 $\pm$ 2.01
	Ours w. ADB	82.34 $\pm$ 2.51	82.89 $\pm$ 1.62	89.49 $\pm$ 0.67	86.36 $\pm$ 0.69	87.63 $\pm$ 2.41	86.77 $\pm$ 2.53
75%	DCL	81.07	86.98	88.08	89.43	81.71	85.85
	ADB	81.08	85.96	86.32	88.53	82.78	85.99
	ODIST	82.79 $\pm$ 1.58	N/A	87.70 $\pm$ 0.74	N/A	83.75 $\pm$ 0.94	N/A
	Ours w/o CC	69.57 $\pm$ 2.56	81.35 $\pm$ 1.05	60.92 $\pm$ 0.86	75.77 $\pm$ 0.50	68.72 $\pm$ 0.74	75.73 $\pm$ 0.30
	Ours w/o GS	80.11 $\pm$ 1.94	84.44 $\pm$ 2.37	88.40 $\pm$ 1.08	90.01 $\pm$ 0.78	84.19 $\pm$ 0.56	87.31 $\pm$ 0.33
	Ours w/o CLS	78.45 $\pm$ 0.41	83.84 $\pm$ 0.51	84.84 $\pm$ 0.33	85.04 $\pm$ 1.19	77.04 $\pm$ 0.54	82.65 $\pm$ 0.47
	Ours	81.81 $\pm$ 1.12	86.11 $\pm$ 1.03	88.28 $\pm$ 0.73	89.78 $\pm$ 0.90	84.23 $\pm$ 1.40	87.23 $\pm$ 1.12
	Ours w. ADB	80.99 $\pm$ 1.82	85.51 $\pm$ 1.21	87.15 $\pm$ 0.63	88.87 $\pm$ 0.63	83.19 $\pm$ 1.19	86.42 $\pm$ 1.11

5 [0135] Table III depicts macro F1-scores of seen versus unseen classes for each of the datasets with the different proportions of seen classes.

[0136] Table III

	Method	Banking		OOS/CLINC		StackOverflow	
		Seen	Unseen	Seen	Unseen	Seen	Unseen
25%	DCL	69.39	80.12	80.43	92.35	63.80	74.86
	ADB	70.94	84.56	76.80	91.84	78.82	90.88
	ODIST	72.72 $\pm$ 1.08	87.11 $\pm$ 2.09	79.69 $\pm$ 2.53	93.42 $\pm$ 1.39	83.18 $\pm$ 2.54	94.41 $\pm$ 1.36
	Ours w/o CC	44.94 $\pm$ 1.38	2.04 $\pm$ 0.60	44.25 $\pm$ 1.51	26.51 $\pm$ 3.07	47.81 $\pm$ 4.44	12.06 $\pm$ 3.29
	Ours w/o GS	70.64 $\pm$ 3.94	83.17 $\pm$ 4.15	82.53 $\pm$ 1.77	94.06 $\pm$ 0.96	62.37 $\pm$ 2.32	67.75 $\pm$ 7.35
	Ours w/o CLS	70.00 $\pm$ 1.29	80.94 $\pm$ 1.15	78.17 $\pm$ 2.80	91.36 $\pm$ 2.43	56.92 $\pm$ 3.27	61.96 $\pm$ 5.29
	Ours	74.83 $\pm$ 2.74	87.08 $\pm$ 2.31	83.01 $\pm$ 1.45	94.30 $\pm$ 0.80	71.15 $\pm$ 3.43	79.67 $\pm$ 4.76
	Ours w. ADB	77.09 $\pm$ 3.43	90.23 $\pm$ 2.06	82.56 $\pm$ 1.88	94.99 $\pm$ 0.62	80.90 $\pm$ 4.07	91.45 $\pm$ 5.62
50%	DCL	79.52	67.26	86.54	90.30	79.22	71.88
	ADB	80.96	78.44	85.00	88.65	87.34	85.68
	ODIST	81.79 $\pm$ 0.81	81.32 $\pm$ 1.54	86.52 $\pm$ 0.87	90.62 $\pm$ 0.71	87.13 $\pm$ 1.41	89.57 $\pm$ 1.04
	Ours w/o CC	62.72 $\pm$ 6.48	4.01 $\pm$ 4.05	62.24 $\pm$ 1.10	19.77 $\pm$ 6.90	66.74 $\pm$ 1.51	21.03 $\pm$ 4.14
	Ours w/o GS	83.32 $\pm$ 1.64	77.61 $\pm$ 4.16	86.59 $\pm$ 1.76	89.29 $\pm$ 1.66	80.47 $\pm$ 5.08	70.78 $\pm$ 14.72
	Ours w/o CLS	79.06 $\pm$ 0.77	73.01 $\pm$ 3.44	81.09 $\pm$ 0.61	87.46 $\pm$ 1.05	73.77 $\pm$ 0.68	60.63 $\pm$ 3.73
	Ours	83.38 $\pm$ 1.29	82.41 $\pm$ 2.08	87.22 $\pm$ 0.77	91.06 $\pm$ 0.82	83.90 $\pm$ 2.01	83.84 $\pm$ 2.33
	Ours w. ADB	81.85 $\pm$ 1.61	83.36 $\pm$ 2.89	86.29 $\pm$ 0.69	91.59 $\pm$ 0.56	86.59 $\pm$ 2.59	88.64 $\pm$ 2.25
75%	DCL	87.47	60.71	89.46	86.28	87.22	65.44
	ADB	86.29	66.47	88.58	83.92	86.80	73.86
	ODIST	87.20 $\pm$ 1.06	71.95 $\pm$ 3.26	89.33 $\pm$ 0.53	85.86 $\pm$ 0.96	87.66 $\pm$ 0.87	75.21 $\pm$ 1.23
	Ours w/o CC	82.12 $\pm$ 1.90	2.70 $\pm$ 2.20	76.28 $\pm$ 0.47	18.76 $\pm$ 4.04	80.14 $\pm$ 0.23	9.48 $\pm$ 1.45
	Ours w/o GS	86.80 $\pm$ 0.31	65.24 $\pm$ 4.24	90.45 $\pm$ 0.77	86.25 $\pm$ 1.43	88.25 $\pm$ 0.48	73.29 $\pm$ 2.70
	Ours w/o CLS	84.23 $\pm$ 0.53	61.17 $\pm$ 3.59	85.05 $\pm$ 1.21	84.77 $\pm$ 0.83	84.66 $\pm$ 0.35	52.48 $\pm$ 2.35
	Ours	86.74 $\pm$ 1.02	69.11 $\pm$ 3.67	89.81 $\pm$ 0.51	86.56 $\pm$ 0.65	88.09 $\pm$ 1.08	74.41 $\pm$ 2.18
	Ours w. ADB	84.29 $\pm$ 1.19	64.92 $\pm$ 3.18	88.89 $\pm$ 0.63	85.90 $\pm$ 0.74	87.39 $\pm$ 1.12	73.29 $\pm$ 1.94

[0137] Top two accuracies are bolded in Tables II and III. Only ODIST provided standard deviations, and “N/A” means that the measure was not provided. The methods “Ours w/o CC”, “Ours w/o GS”, “Ours w/o CLS”, “Ours” and “Ours w. ADB” are ablation studies, representing an embodiment of the present invention without convex combination
5 sampling, an embodiment of the present invention without generic sampling, an embodiment of the present invention without classifier, an embodiment of the present invention (itself), and an embodiment of the present invention with ADB, respectively.

[0138] Tables IV and V show comparisons with the large language model *gpt-35-turbo-16k*. All results of “Ours” are trained with ADB using the base model *roberta-large8*
10 (<https://huggingface.co/roberta-large>), which is believed to be a fairer comparison in terms of cost and speed. Due to the cost associated with the accessing OpenAI’s API, especially for few-shots chain-of-thoughts, results of only a single run is provided

[0139] Table IV depicts the accuracy and F1-scores for LLM-based baselines.

[0140] Table IV

	Method	Banking		OOS/CLINC		StackOverflow	
		Accuracy	F1-Score	Accuracy	F1-Score	Accuracy	F1-Score
25%	ICL-0-shot	36.98	42.90	59.03	56.07	93.28	84.93
	ICL-1-shot	43.31	51.17	68.67	64.13	94.08	88.57
	ICL-3-shot	35.81	48.32	67.62	63.99	89.55	77.18
	CoT-0-shot	33.57	42.77	58.35	58.79	91.25	82.37
	CoT-1-shot	87.11	74.45	93.94	85.71	95.57	91.37
	CoT-3-shot	90.13	79.55	93.89	87.12	95.73	91.64
	Ours (RoBERTa)	86.10 ± 1.59	80.66 ± 1.99	93.34 ± 1.44	86.78 ± 2.36	89.84 ± 2.29	84.80 ± 2.59
50%	ICL-0-shot	46.82	55.09	58.77	67.66	87.35	84.75
	ICL-1-shot	50.91	61.02	71.76	76.37	87.98	85.89
	ICL-3-shot	54.35	64.92	69.74	75.09	89.08	86.88
	CoT-0-shot	48.28	57.54	62.87	71.05	87.12	84.74
	CoT-1-shot	81.95	76.20	88.43	84.95	90.00	87.24
	CoT-3-shot	85.39	81.55	91.02	87.38	90.53	87.39
	Ours (RoBERTa)	84.81 ± 1.09	84.37 ± 1.08	91.19 ± 0.69	88.87 ± 0.77	89.01 ± 2.00	87.54 ± 1.93
75%	ICL-0-shot	53.15	58.48	71.82	72.53	88.12	89.84
	ICL-1-shot	60.81	67.80	79.43	79.65	87.27	89.32
	ICL-3-shot	64.51	71.39	79.09	80.10	87.58	89.56
	CoT-0-shot	55.16	62.13	76.31	76.50	90.07	91.31
	CoT-1-shot	76.04	75.61	86.39	86.38	88.38	89.98
	CoT-3-shot	79.58	77.64	86.52	86.51	89.02	90.59
	Ours (RoBERTa)	84.69. ± 0.76	87.83 ± 0.54	89.67 ± 1.40	90.54 ± 1.40	85.87 ± 1.07	88.31 ± 1.02

[0141] Table V depicts the macro F1-scores of seen versus unseen classes for LLM
baselines.

[0142] Table V

	Method	Banking		OOS/CLINC		StackOverflow	
		Seen	Unseen	Seen	Unseen	Seen	Unseen
25%	ICL-0-shot	43.13	38.56	55.84	64.50	82.76	95.73
	ICL-1-shot	51.51	44.70	63.78	77.02	87.05	96.18
	ICL-3-shot	49.36	28.59	63.71	74.80	73.95	93.29
	CoT-0-shot	43.37	31.31	58.61	65.75	79.96	94.39
	CoT-1-shot	73.53	92.02	85.43	96.20	90.22	97.12
	CoT-3-shot	78.79	93.99	86.88	96.25	90.52	97.25
	Ours (RoBERTa)	80.05 $\pm$ 1.02	90.26 $\pm$ 1.49	85.50 $\pm$ 2.41	95.78 $\pm$ 1.92	81.56 $\pm$ 3.74	91.98 $\pm$ 2.87
50%	ICL-0-shot	46.82	55.09	58.77	67.66	84.28	89.41
	ICL-1-shot	50.91	61.02	71.76	76.37	85.52	89.52
	ICL-3-shot	54.35	64.92	69.74	75.09	86.46	91.12
	CoT-0-shot	48.28	57.54	62.87	71.05	84.22	89.86
	CoT-1-shot	81.95	76.20	88.43	84.95	86.75	92.18
	CoT-3-shot	85.39	81.55	91.02	87.38	86.79	93.32
	Ours (RoBERTa)	84.81 $\pm$ 1.09	84.37 $\pm$ 1.08	91.19 $\pm$ 1.69	88.87 $\pm$ 1.77	88.91 $\pm$ 2.33	90.85 $\pm$ 1.50
75%	ICL-0-shot	53.15	58.48	71.82	72.53	90.39	81.66
	ICL-1-shot	60.81	67.80	79.43	79.65	89.92	80.25
	ICL-3-shot	64.51	71.39	79.09	80.10	90.13	81.06
	CoT-0-shot	55.16	62.13	76.31	76.50	91.72	85.19
	CoT-1-shot	76.04	75.61	86.39	86.38	90.48	82.50
	CoT-3-shot	79.58	77.64	86.52	86.51	91.07	83.27
	Ours (RoBERTa)	84.69 $\pm$ 1.76	87.83 $\pm$ 1.54	89.67 $\pm$ 1.40	90.54 $\pm$ 1.40	89.91 $\pm$ 2.01	87.43 $\pm$ 2.14

B-I. Overview

[0143] Across the nine test cases, the proposed method 400 (“Ours”), which may be described in similar context to the method 100 of FIG. 1, produces competitive or better results with the previous best results on seven cases, except on the 25% and 50% proportions of the StackOverflow dataset. In these two cases, augmenting the proposed method with ADB (“Ours w. ADB”) further improves the results to near competitive levels.

B-II. Dataset Comparisons

[0144] The results show that prompting performs relatively better on the CLINC dataset than the Banking dataset, achieving an average 7.99% relative gain in accuracy over ODIST on CLINC versus 0.9% on Banking. In particular, a 14.8% relative gain on the 25 proportion of the CLINC dataset has been observed. This is likely caused by the nature of the two datasets. Banking is a fine-grained dataset on a single domain, so that vocabularies between classes are often similar. While the generic sampling (as described in the context of the outlier samplers used, 344 of FIG. 3) helps with unbiasing this, the language modeling nature of prompt-based approaches makes the Banking dataset inherently a more challenging dataset than CLINC.

[0145] Several plausible reasons behind the weaker performance on the StackOverflow dataset have been contemplated, in hope that future work may improve upon these. The first comes from a geometric interpretation in the latent space. All methods listed in Tables **[0146]** II and III aim to fill the entire latent space with out-of-domain samples, barring the area around the centroid of each seen intent class. By using convex combinations, for the n -th seen intent, an additional $(n - 1)$ may construct “areas” of out-of-domain samples, and the total coverage grows in the order of $O(n^2)$ with respect to the number of classes. Therefore, for the StackOverflow dataset, where the number of classes are significantly smaller (i.e. with only 20), convex combinations alone are not sufficient to detect out-of-domain samples.

[0147] The size of the dataset likely also plays a role. Referring to Table I, it can be seen that StackOverflow has 1,000 samples per class, compared with roughly 160 for the remaining two. Both ADB and ODIST rely more heavily on these seen intent samples – ADB learns a decision boundary around these samples, whereas ODIST takes the samples and directly replaces the text, and thus it is more likely to benefit from having more in-domain intent samples. This is in contrast to the proposed method 400, which depends more on the number of seen intents classes as described above, and is likely a strong reason ADB and ODIST perform significantly better on StackOverflow, the dataset with a much higher sample per class ratio.

[0148] The final is the technical language of the dataset. This means that by virtue of the byte-pair encoding used by most language models (e.g. BERT uses WordPiece), many of the answers, such as “osx”, are not native to the tokenizer’s vocabulary. This in turns means that these labels are split into multiple meaningless tokens, and makes it more challenging for the language model to leverage the knowledge from the pretraining corpus. In these cases, prompting loses the advantage over the one-hot encoding used in supervised models, since some of the labels do not have actual semantic meaning.

B-III. Baseline Comparisons

[0149] Based on Tables II and III, the results with respect to each of the given baseline are analyzed, highlighting the strengths and weaknesses of each of the methods.

[0150] The proposed method 400 (FIG. 4) is compared with DCL, which shares some of the same outlier sampling method, but adds external datasets on top of synthesizing OOD samples. The results of the proposed method 400 (denoted as “Ours”), even without the additional ADB layer and any external datasets, are far more consistent across all the different datasets and proportions. This gain is particularly noticeable in the datasets with fewer intents, with relative gains of 23.0% and 23.4% in Banking and StackOverflow, respectively. Given the similarities in the underlying algorithms of these two methods, this is a strong evidence to suggest that the synthesizing outliers in terms of prompts is far more efficient than under the fully-supervised situation.

[0151] It has also been observed that applying ADB on prompting is an improvement over ADB per se in the vast majority of the cases with up to 35.9% relative gain in accuracy (comparing “Ours w. ADB” and “ADB”). This shows that employing prompt may address some of the weaknesses of ADB. Since ADB trains a spherical boundary around each inlier class, this implicitly determines the geometry of the boundary, which may not be representative on all datasets. Its performance may also degrade if there are not sufficient samples per seen class. On the other hand, the proposed method 400 does not make any assumption on the decision boundary between any of the inlier or outlier classes, and thus solves this problem when used along with ADB. It is also advantageously less dependent on the intent per class ratio, which again pairs well to be used alongside with ADB.

[0152] However, interestingly it does not always yield major improvements over the prompting results (i.e. comparing “Ours” and “Ours w. ADB”). In general, it helps significantly on all three 25% proportions, similar or slightly better results on the 50% proportions, and slightly worse performances on the 75% proportions. A possible explanation may be that prompting does not separate the classes as distinctly as a fully-supervised model, since class labels in the same domain with similar vocabularies inevitably share some degree of similarity. This makes ADB, which is a distance-based boundary learning algorithm, less likely to perform well on the proposed trained models, especially as the proportion increases.

[0153] A similar comparison may be made with ODIST. Since it directly modifies in-domain data on the word level to generate outlier samples, it is equally reliant on the number of samples per class in the dataset. A similar trend is therefore observed, where the

proposed method 400 (“Ours”) is competitive or superior to ODIST (“ODIST”) in five of the six settings for Banking and CLINC, but inferior in two of the three settings in StackOverflow. It is also important to note that when the number of classes is high enough, the proposed method 400 may even match that of ODIST in the 75% proportion of the StackOverflow dataset. While the proposed method 400 may be integrated with ODIST in principle, ADB has been chosen due to its simplicity and less demand on resources as ODIST requires a separate language model during training.

[0154] Some fundamental differences of the different approaches – generative training versus boundary learning may suggest that the results from the proposed method 400 may be stronger compared with ODIST and ADB in the cases where the dataset is larger (in terms of the number of classes, e.g. CLINC), but also weaker when the dataset is smaller (e.g. StackOverflow), although this weakness may be addressed by augmenting with ADB. Learning a boundary may be viewed as minimizing the radii of the decision boundaries of the known intent, given the same known samples. Its success is therefore mostly dependent on the quality of the samples within the same class, and less on the number or distributions of other classes. On the other hand, generative methods, like the proposed method 400 and DCL, aim to maximize the boundary of some unseen intent, but based on the seen intents. Therefore, the proposed method 400 as well as DCL, as evident by their respective results, may struggle to compete with ADB or ODIST when the dataset is too small to draw comprehensive representations of the unseen intent. However, given enough class samples, both generative methods are able to match and even outperform boundary learning methods.

B-IV. Comparison with Other Prompting Techniques with LLMs

[0155] Amongst the nine settings, the proposed model significantly outperforms chain-of-thought with LLM in two settings (Banking 75% and OOS 75%), significantly underperforms in one setting (StackOverflow 25%) and is competitive for the remaining six. It should also be noted that chain-of-thought prompting with LLMs tends to cause it to “over-classify” samples as out-of-domain, leading to a high accuracy but low (macro) F1-Score for the lower percentage settings; the proposed models are in general more balanced.

[0156] In particular, it has been observed that chain-of-thought with LLM struggles more on the Banking dataset, which is again unsurprising given that the intent classes and examples are syntactically very similar for that dataset. This also explains the huge difference between 0-shot and 1-shot results, where the chain-of- thought examples tell the language model explicitly why the utterances are classified into their respective categories.

[0157] On the contrary, GPT3.5 performs exceptionally well on the StackOverflow dataset, even at zero shot settings. This may be attributed – based on observations on GPT3.5’s ability to code – to the fact that it has most likely been explicitly trained (either during pre-training, or via reinforcement learning (<https://openai.com/blog/chatgpt>)) on similar domain data or code repositories.

B-V. Scaling with Training Data

[0158] Additional experimental results are provided on how the proposed method scales when the amount of training data varies. FIGS. 5, 6 and 7 shows the plots of the amount of training data versus the macro-F1 scores achieved by the proposed model and the various baselines (DCL, ABD, ODIST) for the CLINC dataset, at the proportions of 25%, 50%, and 75%, respectively. Each data point on FIGS. 5 to 7 is the average of three runs. Overall, these plots show that the gains observed in Tables II and III for the full training set also translates to low resource settings.

[0159] In particular, while the performance of all the models are consistent for higher training percentages, significant drops in accuracy for baselines such as ODIST in some of the lowest data percentages have been observed. On the contrary, it is noted that the proposed model performs notably better in these cases, boasting an average relative gain of about 20.5% over ODIST across the three test cases, when restricting the training data to about 10%. This is a strong evidence to suggest that data generation via prompting is far more data efficient, and fewer data are required for the model to learn a good representation of OOD samples.

B-VI. Robustness Analysis

[0160] Prompting techniques are known to be sensitive to the choice of exemplars. FIG. 8 shows a plot that compares the means and standard deviations of the various prompts

P_1 to P_5 described in the Prompt Reformation Section for the 50% proportion of the Banking dataset. It is observed that the proposed method 400 is fairly robust overall. Ensemble, used in all the results of the proposed method 400, is done by taking majority votes of the individual prompts. While the variance of results for some other prompting methods may range from near state-of-the-art to near randomness, in general, each individual prompt performs very similarly. This is perhaps unsurprising since the proposed model is trained in a supervised-manner, so the exemplar choice is much less significant than any in-context learning-based approaches. By doing prompt ensemble, the standard deviation of the results of the proposed method 400 may be reduced, further strengthening the overall robustness.

B-VII. Ablation Studies

[0161] The most crucial component may be the generation of convex combinations, without which models are not able to distinguish outliers at all. But it is the other two ablation tests that shared some insights towards the usefulness and limitations of prompting.

[0162] Without generic sample (“Ours w/o GS”), the drop in accuracy varies more widely across the datasets. CLINC, the dataset where the classes are most separated in terms of vocabulary, sees by far the smallest impact of generic samples, with a maximum of 15% drop in average accuracy in the three settings. On the other hand, the Banking and StackOverflow datasets both see a maximum of 25% decrease in accuracy, which is again more apparent with the less seen proportions. This is mostly in line with the intuition behind generic sampling, which helps fill the latent space with OOD samples when the number of seen intents are too small to cover it with convex combinations. This also explains the phenomenon that at the 75% settings, generic samples are largely made redundant as the amount of convex combinations along is able to provide the model with enough OOD samples.

[0163] Models also fail to reach the same level of performance without a classifier head (“Ours w/o CLS”). It should be noted that in these runs, stopwords have not been removed from the class labels since they cause some of the tags to be identical. A significant drop in performance is observed in all test cases this time, rather than limited to certain dataset.

This highlights the limitations of prompting when compared with fully-supervised models (at least in the proposed method 400). Since prompting essentially learns to predict the text labels themselves – rather than the semantically meaningless one-hot encodings, it may naturally struggle more when some of the class labels are semantically similar (such as

5 `topping_up_by_card` and `top_up_failed` in the Banking dataset). On the contrary, fully supervised learning relies on separating classes in the latent embedding space, and hence when it is told that these classes are different, it separates them more easily. Different methods of answers prompt designs may be explored in the future to see whether this boosts the performance to the same level as fully-supervised models.

10 **[0164]** While the invention has been particularly shown and described with reference to specific embodiments, it should be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention as defined by the appended claims. The scope of the invention is thus indicated by the appended claims and all changes which come within the meaning and

15 range of equivalency of the claims are therefore intended to be embraced.

CLAIMS

Claim 1: A method for performing intent detection and classification, the method comprising:

- 5 receiving an input text;
- reformulating the input text as one or more prompts;
- passing the one or more prompts through a language model to obtain one or more states based on the input text, wherein the language model comprises a classifier head and a language modeling head configured to operate in parallel with the classifier head;
- 10 based on the one or more states, predicting a classifier output by the classifier head, and simultaneously generating a language modeling prediction by the language modeling head; and
- combining the classifier output and the language modeling prediction to generate an output prediction, wherein the output prediction is out-of-distribution if the language
- 15 modeling prediction is indicative that the one or more states are predicted to be outside an in-domain label space, or otherwise, the output prediction is an in-distribution intent, wherein the classifier output comprises the in-distribution intent falling within the in-domain label space.

- 20 Claim 2: The method as claimed in claim 1, wherein
- each of the one or more prompts comprises an input slot and an answer slot,
- reformulating the input text as the one or more prompts comprises filling or replacing the input slot with the input text, and
- the method further comprises:
- 25 mapping the output prediction to a set of tokens;
- determining a token with a highest average probability or a majority voting from the set of tokens; and
- filling the answer slot with the determined token.

- 30 Claim 3: The method as claimed in claim 2, wherein the one or more prompts comprise at least one of the following:

a first comment about the input text and a first question concerning the token;
a second comment about the input text and a second question concerning the token,
the second comment being different from the first comment and the second question being
different from the first question;

5 a third comment about the input text and a topic concerning the token;
 a first query about the input text and a user enquiry about the token; or
 a second query about the input text and an enquiry about the token, the first query
being different from the second query and the user enquiry being different from the
enquiry.

10

Claim 4: The method as claimed in any one of claims 1 to 3, wherein reformulating
the input text as one or more prompts further comprises pre-processing the input text.

Claim 5: The method as claimed in claim 4, wherein pre-processing the input text
15 comprises removing a non-alphanumeric character and/or a stopword originating from the
input text.

Claim 6: The method as claimed in any one of claims 1 to 5, wherein the language
model further comprises a pre-trained model in upstream communication with the classifier
20 head and the language modeling head, and the one or more states comprise one or more
hidden states generated by the pre-trained module.

Claim 7: The method as claimed in claim 6, wherein the pre-trained model comprises
a Bidirectional Encoder Representation from Transformer (BERT), or a Generative pre-
25 trained transformer (GPT), or a Pathways Language Model (PaLM), or a Large Language
Model Meta AI (Llama), or Efficiently Learning an Encoder that Classifies Token
Replacements Accurately (ELECTRA), or Generalized Autoregressive Pretraining for
Language Understanding (XLNet), a Robustly Optimized BERT Pretraining Approach
(RoBERTa), or a Decoding-enhanced BERT with Disentangled Attention (DeBERTa).

30

Claim 8: The method as claimed in any one of claims 1 to 7, wherein the language model is jointly trained with a classification task and a language modeling task, and using at least two outlier samples, wherein one of the at least two outlier samples is generated by a first technique and another of the at least two outlier samples is generated by a second technique.

Claim 9: The method as claimed in claim 8, wherein the first technique comprises performing a convex combination of a first embedding (h_x) associated with a first training input (x) and a second embedding ($h_{x'}$) associated with a second training input (x'), the first training input and the second training input being from different classes, and wherein an embedding (h) of the outlier sample generated by the first technique is represented by:

$$h = \theta h_x + (1-\theta) h_{x'}$$

where a distribution of θ is a uniform distribution given by $\theta \sim U(0, 1)$.

Claim 10: The method as claimed in claim 8 or 9, wherein the second technique comprises:

computing a mean value (μ) and a standard deviation (σ) of all sentence lengths in a training set of the in-domain label space;

removing stopwords from the training set to construct a set of words (W); and

uniformly sampling from a uniform distribution of the set of words (W) to form the outlier sample comprising a generic sample represented by:

$$x_i \sim U(W) \text{ for } 0 < i \leq l$$

where $l \sim N(\mu, \sigma^2)$.

Claim 11: The method as claimed in any one of claims 8 to 10, wherein the classification task comprises:

applying, by the classifier head, a softmax function on a plurality of hidden states to obtain classifier logits, wherein the plurality of hidden states is generated by passing training prompts reformulated from a training input text of a specific class through the language model; and

training the classifier head with cross entropy loss based on the classifier logits.

Claim 12: The method as claimed in claim 11, wherein the language modeling task comprises:

5 applying, by the language modeling head, a conditional softmax function on the plurality of hidden states to obtain language modeling logits; and
training the language modeling head with binary cross entropy loss based on the language modeling logits.

10 Claim 13: The method as claimed in any one of claims 8 to 12, wherein the at least two outlier samples are trained with binary cross entropy loss.

Claim 14: The method as claimed in claim 13, wherein the language model in entirety, upon training, has a weighted loss function comprising a summation of the cross entropy loss based on the classifier logits, a weighted binary cross entropy loss based on the
15 language modeling logits, and a weighted binary cross entropy loss associated with the at least two outlier samples.

Claim 15: The method as claimed in any one of claims 1 to 14, wherein the language model is post-trained with an adaptive decision boundary.

20

Claim 16: A system for performing intent detection and classification using a language model comprising a classifier head and a language modeling head configured to operate in parallel with the classifier head, the system comprising:

25 a memory; and
at least one processor unit in communication with the memory, the at least one processor unit configured to:
receive an input text,
reformulate the input text as one or more prompts,
pass the one or more prompts through the language model to obtain one or
30 more states based on the input text, and

combine a classifier output and a language modeling prediction to generate an output prediction,

wherein based on the one or more states, the classifier head is configured to predict the classifier output, and simultaneously, the language modeling head is configured to generate the language modeling prediction; and

wherein the output prediction is out-of-distribution if the language modeling prediction is indicative that the one or more states are predicted to be outside an in-domain label space, or otherwise, the output prediction is an in-distribution intent, wherein the classifier output comprises the in-distribution intent falling within the in-domain label space.

Claim 17: The system as claimed in claim 16,

wherein each of the one or more prompts comprises:

an input slot for filling the input text; and

an answer slot,

wherein the system further comprises a verbalizer configured to map the output prediction to a set of tokens, and

wherein the at least one processor unit is further configured to determine a token with a highest average probability or a majority voting from the set of tokens and fill the answer slot with the determined token.

Claim 18: The system as claimed in claim 16 or 17, wherein the at least one processor unit is further configured to remove a non-alphanumeric character and/or a stopword originating from the input text.

Claim 19: The system as claimed in any one of claims 16 to 18, wherein the language model further comprises a pre-trained model in upstream communication with the classifier head and the language modeling head, the pre-trained model being configured to generate the one or more states.

Claim 20: The system as claimed in claim 19, wherein the pre-trained model comprises a Bidirectional Encoder Representation from Transformer (BERT), or a Generative pre-trained transformer (GPT), or a Pathways Language Model (PaLM), or a Large Language Model Meta AI (Llama), or Efficiently Learning an Encoder that Classifies Token Replacements Accurately (ELECTRA), or Generalized Autoregressive Pretraining for Language Understanding (XLNet), a Robustly Optimized BERT Pretraining Approach (RoBERTa), or a Decoding-enhanced BERT with Disentangled Attention (DeBERTa).

Claim 21: The system as claimed in any one of claims 16 to 20, wherein the language model is jointly trained with a classification task and a language modeling task, and using at least two outlier samples.

Claim 22: The system as claimed in claim 21, wherein during training with the classification task, the classifier head is configured to apply a softmax function on a plurality of hidden states to obtain classifier logits, and configured to be trained with cross entropy loss based on the classifier logits, wherein the plurality of hidden states is generated by passing training prompts reformulated from a training input text of a specific class through the language model.

Claim 23: The system as claimed in claim 22, wherein during training with the language modeling task, the language modeling head is configured to apply a conditional softmax function on the plurality of hidden states to obtain language modeling logits, and configured to be trained with binary cross entropy loss based on the language modeling logits.

Claim 24: The system as claimed in any one of claims 21 to 23, wherein the at least two outlier samples are configured to be trained with binary cross entropy loss.

Claim 25: The system as claimed in claim 24, wherein the language model in entirety, upon training, has a weighted loss function comprising a summation of the cross entropy loss based on the classifier logits, a weighted binary cross entropy loss based on the

language modeling logits, and a weighted binary cross entropy loss associated with the at least two outlier samples.

Claim 26: The system as claimed in any one of claims 16 to 25, wherein the language
5 model is configured to be post-trained with an adaptive decision boundary.

Claim 27: A method for training a language model for intent detection and classification, the method comprising:

jointly training the language model with a classification task and a language
10 modeling task; and

using at least two outlier samples, wherein one of the at least two outlier samples is generated by a first technique and another of the at least two outlier samples is generated by a second technique different from the first technique.

15 Claim 28: The method as claimed in claim 27, wherein the first technique comprises performing a convex combination of a first embedding (h_x) associated with a first training input (x) and a second embedding ($h_{x'}$) associated with a second training input (x'), the first training input and the second training input being from different classes, and wherein an embedding (h) of the outlier sample generated by the first technique is represented by:

20
$$h = \theta h_x + (1-\theta) h_{x'}$$

where a distribution of θ is a uniform distribution given by $\theta \sim U(0, 1)$.

Claim 29: The method as claimed in claim 27 or 28, wherein the second technique comprises:

25 computing a mean value (μ) and a standard deviation (σ) of all sentence lengths in a training set of an in-domain label space;

removing stopwords from the training set to construct a set of words (W); and
uniformly sampling from a uniform distribution of the set of words (W) to form the outlier sample comprising a generic sample represented by:

30
$$x_i \sim U(W) \text{ for } 0 < i \leq l$$

where $l \sim N(\mu, \sigma^2)$.

Claim 30: The method as claimed in any one of claims 27 to 29, wherein the classification task comprises:

5 applying, by a classifier head of the language model, a softmax function on a plurality of hidden states to obtain classifier logits, wherein the plurality of hidden states is generated by passing training prompts reformulated from a training input text of a specific class through the language model; and

training the classifier head with cross entropy loss based on the classifier logits.

10 Claim 31: The method as claimed in claim 30, wherein the language modeling task comprises:

applying, by a language modeling head of the language model, a conditional softmax function on the plurality of hidden states to obtain language modeling logits; and

15 training the language modeling head with binary cross entropy loss based on the language modeling logits, the language modeling head being configured to operate in parallel with the classifier head.

Claim 32: The method as claimed in any one of claims 27 to 31, wherein using the at least two outlier samples comprising training the at least two outlier samples with binary
20 cross entropy loss.

Claim 33: A non-transitory computer-readable storage medium comprising computer readable instructions operable when executed by a computer to perform intent detection and classification, the computer readable instructions configured to perform a method as
25 claimed in any one of claims 1 to 15 and/or a method as claimed in any one of claims 27 to 32.