

Method and Apparatus for Multi-label Text Classification

Field

- 5 The present invention relates to a method and an apparatus for multi-label text classification. In particular, the present invention relates to grammar model based hierarchical semantic analysis for natural language processing of text to be subject to multi-label text classification involving a pre-trained language model and a supervised learning model for text classification.

10 Background

- At present, digitalization and internet of things create huge volume of data, including text data. Automatic target information extraction and tagging from text data are much required in a wide range of industries for key information collection and detection, decision-making, customer
15 feedback and product analysis, as well as future trend forecasting. However, to enable a computer or machine to properly segment human's flexible expressions, identify target information and understand the meaning of the key information is still very challenging.

- The core technology behind target information extraction and tagging is text multi-label
20 classification. Current text data classification mostly relies on "embedding-similarity measurement" methods, which has two categories: Supervised learning model-based embedding and similarity measurement, and pre-trained language model-based unsupervised learning and similarity measurement (See Fig. 6). The main problems with these current methods are:

- a) it usually requires a large volume of high-quality training data and long training process, or
25 b) the accuracy of information extraction and classification varies with different types of description.

- In particular, semantic multi-label tagging with dynamic labels for a given text in natural language has always been a challenging problem. Although various conventional multi-label classification
30 methods or tools have been proposed, the following challenges still exist in seeking to achieve accurate, efficient, and flexible multi-label text classification.

Most existing methods directly take the whole text (short paragraph or sentence) as input, and then perform word tokenization based on them, which may fail to capture semantic similarity

effectively. Some methods are focused on binary relevance, which ignores correlation between labels by treating each label independently. Some methods are based on a label powerset, which has high computational complexity and leads to an imbalanced dataset with multiple classes associated with few examples. Some methods are based on Multi-label k Nearest Neighbours (MLkNN), which relies on binary relevance strategy and ratio estimation, and data distribution for some labels are imbalanced. Furthermore, some method are based on a classifier chain, in which accuracy heavily depends on the chosen ordering of labels, and for n labels, there are n possible orderings.

Furthermore, most methods or tools for multi-label classification focus on algorithm performance and lack a concrete preprocessing pipeline for raw data, such as those used in product feedback analysis, which hinders the application of these multi-label classification methods.

Some commercial solutions have been developed with user-friendly GUI, which are augmented with comprehensive data-preprocessing procedure like MonkeyLearn. However, they are still inadequate for semantic multi-label text classification, as they do not support rapid label learning, real-time label assembly update, or grammar model-based analysis. Taking real-time label assembly update as an example. In many cases, although a large corpus might be used to train the machine learning model, it is still possible that a particular phrase or sentence in a customer feedback may be missed out, especially given that the customers might be unaware of the pre-defined labels or tags when they submit the feedback.

Overall, it is still very challenging for the state-of-the-art methods to solve the following challenges and problems:

- 1) the problem that requires instant multi-label text classification without a tedious training process;
- 2) the need for dynamic label assembly;
- 3) High-dimensional multi-label text classification; and
- 4) High accuracy for multi-label text classification.

Summary

According to an example of the present disclosure, there are provided a method and an apparatus for multi-label text classification as claimed in the independent claims. Some optional features are defined in the dependent claims.

Brief Description of the Drawings

5 Examples in the present disclosure will be better understood and readily apparent to one skilled in the art from the following written description, by way of example only and in conjunction with the drawings, in which:

Fig. 1 is a flowchart illustrating a method according to an example of the present disclosure.

10 Fig. 2 illustrates components of a system or apparatus according to an example of the present disclosure.

Fig. 3 is a block diagram illustrating an apparatus according to an example of the present disclosure that comprises a new Rapidly adapted text tagging engine.

15 Fig. 4 illustrates a hybrid multi-label text classification model according to an example of the present disclosure configured to predict labels for short text and compare probabilities of the predicted labels being the right labels to filter out predicted labels with high probabilities.

20 Fig. 5 shows a conventional text mining engine that is different from a text mining engine according to an example of the present disclosure.

Fig. 6 shows conventional supervised learning and unsupervised learning for embeddings-similarity measurement based text classification that are different from a hybrid classification
25 model according to an example of the present disclosure.

Fig. 7 shows a flowchart illustrating a customer feedback analysis process, which an apparatus according to an example of the present disclosure can be used in such process.

30 Fig. 8 is a block diagram showing a system architecture according to an example of the present disclosure.

Detailed Description

Examples of a comprehensive, hierarchical grammar model-based method are discussed in the present disclosure to realize semantic multi-label text tagging with a dynamic labelling system (label set) .

In one example, firstly, a hierarchical Paragraph2Chunk text analysis structure is built to extract target text snippets by decomposing a paragraph into different levels of text chunks. The Paragraph2Chunk structure, which is scalable and customizable, comprises of hierarchical decomposing functions to breakdown paragraphs, complex sentences, sentences, and word snippets. Secondly, customizable dependency and constituency grammatical models are built and embedded to levels of the Paragraph2Chunk structure for accurate, semantic identification of key contextual information. Instead of text analysis-based on word or sentence embeddings, the grammar model provides semantic and even logic text understanding. Thirdly, a "hybrid" multi-label text classification model is built to achieve accurate semantic multi-label classification by integrating quick supervised learning structure with character-level bag-of-words and unsupervised label prediction with bidirectional learning model.

The example provides an automatic text mining system (or engine) for facilitating rejected product analysis. To build such automatic text mining system, a text Artificial Intelligence (AI) engine and three software modules are developed for multi-label tagging with a customizable and high dimension labelling system. This text mining system includes multi-language processing, and semantic product problem identification, and this text mining system overcomes the issues of complicated supervised training and learning processes that are commonly present in current text mining systems.

The challenges of existing technology for semantic multi-label tagging with dynamic labels for a given text in natural language are discussed below.

(A) Most existing methods directly take the whole text (short paragraph or sentence) as input, and then perform word tokenization based on them, which may fail to capture semantic similarity effectively. A summary of the issues with existing methods involving Binary Relevance, Label Powerset, Multi-label k Nearest Neighbours (MLkNN), and Classifier Chain is provided below.

Binary Relevance:

Ignores correlation between labels by treating each label independently.

Label Powerset:

- i) High computational complexity; and
- 5 ii) Leads to imbalanced dataset with multiple classes associated with few examples.

Multi-label k Nearest Neighbours (MLkNN):

- i) Relies on binary relevance strategy and the ratio estimation; and
- ii) Data distribution for some labels are imbalanced.

Classifier Chain:

- 10 i) Accuracy heavily depends on the chosen ordering of labels; and
- ii) for n labels, there are n possible orderings.

(B) A single method may be less effective in handling different levels of input segments, such as chunks or characters.

15

(C) Dependency grammar helps to understand the content in each text by considering the relations between the individual words in the text. However, most existing methods do not explicitly consider it.

20

(D) Most methods or tools focus on algorithm performance and lack a concrete preprocessing pipeline for the raw data, such as those used in product feedback analysis, which hinders the application of these multi-label classification methods.

25

(E) The underlying models (Sentence-BERT and bart-large-mnli) used in pre-trained language model-based approaches are particularly sensitive to the semantic structure of the text and certain words in the text, as they are pre-trained in a self-supervised manner based on masked language modelling. Thus, they are unable to disambiguate between different labels with highly similar semantic structures effectively.

30

(F) The above approaches set out in paragraphs (A) to (D) are generally only performed on multiclass text classification tasks. Although these approaches can be readily extended to multi-label text classification tasks by selecting labels where cosine similarities/probability scores exceed a certain threshold, the text could be wrongly classified with certain labels when there are labels with highly similar semantic structures and yet are opposite in

meaning, or there may be missing labels as the cosine similarity/probability score for the correct label may not cross a pre-fixed threshold.

5 (G) Embedding-based supervised learning relies heavily on the quality of processed training data and diversity of trained features. This may lead to high costs of model maintenance.

10 (H) Although some commercial solutions have been developed with user-friendly Graphical User Interface (GUI), which are also augmented with comprehensive data-preprocessing procedure like MonkeyLearn, they are still inadequate for semantic multi-label text classification, as they do not support rapid label learning, real-time label assembly update, or grammar model-based analysis. Taking real-time label assembly update as an example, in many cases, although a large corpus might be used to train the machine learning model, it is still possible that they may miss out on a particular phrase or sentence in a customer feedback, especially given that the customers might be unaware of the pre-defined labels
15 or tags when they submit the feedback. Therefore, allowing to update the label assembly for the model training in a real time fashion would be important for a desirable solution.

Overall, it is still very challenging for the state-of-the-art methods to solve the following problems:

- 20 1. The problem that requires instant multi-label text classification without a tedious training process.
2. The need for a dynamic label assembly.
3. High-dimensional multi-label text classification.
4. High accuracy for multi-label text classification.

25 In an example of the present disclosure, three important text analysis models are proposed and created for a system or apparatus. Firstly, a hierarchical Paragraph2Chunk (Paragraph-to-Chunk) text analysis structure is created. Such "Text chunking" process in Natural Language Processing (NLP) involves the conversion of unstructured text data into meaningful units called "chunks". The Paragraph2Chunk structure is a scalable text analysis structure where the Paragraph-Complex
30 Sentence-Simple Sentence-Clause-Chunk forms the base structure for a semantic, logic text analysis. Secondly, customizable dependency and/or constituency grammatical models are built and applied to levels of Paragraph2Chunk to realize automatic segmentation of target text information. Lastly, with the "chunks" generated from the grammar model, a multi-label text classification model is created with character-level bag-of-words (or character-level n-gram

embedding) training (i.e. involving a character-level bag-of-words model) and a pre-trained language model (e.g. SBERT may be used). This “hybrid” multi-label text classification model achieves the best results in terms of both accuracy and efficiency when compared with state-of-art methods for multi-label text classification. The character-level bag-of-words model is a type of supervised learning model for text classification with fast learning ability.

Fig 1. shows a flowchart illustrating a method 100 of Paragraph2Chunk based semantic multi-label classification that can be performed by the aforementioned example. As shown in Fig. 1, the base form of the Paragraph2Chunk structure includes levels of Paragraph (P), Complex Sentence (CS), Simple sentence (SS), Clause (CLS) and Chunk (CK). The P level involves input text data. If the input text data has more than one paragraphs of text, it can be split into sentences based on punctuation and customized characters at a step 102. For example, customized characters may include spacings between paragraphs, bullet points and numbering indicative of separate items in a list, capital letters indicative of the start of a sentence, etc. Punctuation such as full stop can indicate end of a sentence. Examples of customized characters include newline characters '\r', '\n' that indicate spaces between paragraphs, parts such as '1.', '2', 'Clause:' (word or words followed by a colon), etc. indicative of separate items in a list. The input text data can be in the English language, Chinese language, or other compatible language.

The CS level comprises split sentences from step 102. The split sentences may comprise complex sentences and simple sentences. Detection of a complex sentence may involve detection of one independent clause and at least one dependent/subordinate clause.

Generally, a dependency grammar model makes use of dependency parsing that focuses on identifying the grammatical relationships between words in a sentence, such as subject-verb relationships. A constituency grammar model makes use of constituency parsing that focuses on identifying the constituent structure of a sentence, such as noun phrases and verb phrases.

A customized dependency and/or constituency grammar model can be used to split or extract target text at a step 104. In the present example, specifically, a dependency grammar model is used at step 104.

The customized dependency grammar model used at step 104 may be developed to analyse and infer both structure and semantic dependencies and relationships between tokens in a sentence.

One or more dependency trees representing these dependencies and relationships may be used for information (i.e. clause) retrieval for the CLS level processing, which involves Part of Speech (POS) segmentation at a step 106.

5 In the one or more dependency trees used, labelled edges depict dependency relationships between heads and their dependents. A word may be selected as an overall head of a phrase and other words can be selected as dependents of the head. The labels on the edges indicate the grammatical role of the dependent. The idea behind a dependency grammar is that in any sentence in the language, all words except one have some relationship or dependency on other words in the sentence. The word that has no dependency is called the root of the sentence. The verb is taken as the root of the sentence in most cases. All the other words are directly or indirectly linked to the root verb using links, which are the dependencies. If a dependency syntax tree is to be drawn for a sentence in the inputted text data, the dependency relationships would each have its own meanings and be part of a list of universal dependency types.

15 Dependencies are labelled word-word relations or links that are usually asymmetrical. A word has a relation or depends on another word based on the positioning of the words in the sentence. Consequently, dependency grammars assume that further constituents of phrases and clauses are derived from this dependency structure between words. As mentioned, the word that has no dependency is termed the root of the sentence. The verb is taken as the root of the sentence in most cases. All the other words are directly or indirectly linked to the root verb using links, which are the dependencies. In the case that there are no concepts in the phrases or clauses, looking at the syntax and relations between words and their dependents may allow one to determine the necessary constituents in the sentence.

25 In the dependency grammar model for step 104, there is a one-to-one relationship to each word in the sentence. There are two aspects to this grammar representation. One is the syntax or structure of the sentence and the other is the semantics obtained from the relationships denoted between the words. The syntax or structure of the words and their interconnections can be shown using a sentence syntax or parse tree.

30 In the example of Fig. 1, all sentences (including both complex and simple sentences) will be broken down into simple sentences (SS level) by step 104. A simple sentence may comprise just one independent clause in the sentence. A simple sentence may also relate to just one subject

and has only a single verb. Part-of-speech (POS) based segmentation will be performed on each simple sentence to extract clauses (CLS level) at step 106. The clauses may include short sentences, which may be incomplete. Examples A and B below illustrate how complex sentences can be broken down into clauses.

5

Example A:

Text (which is a complex sentence): "water leaks and a lot of steam comes out, so the cover is very wet"

List of short sentences: "water leaks and a lot of steam comes out", "so the cover is very wet"

10 List of clauses: "water leaks", "and a lot of steam comes out", "so the cover is very wet"

In example A, there are two short sentences split from the complex sentence along the single punctuation character ",", the second of which contains just a single clause. The first short sentence is further split into two clauses based on the conjunction "and".

15

Example B:

Text: "water leaks"

In example B, there is only one short sentence and one clause arising from this text. Complex sentence processing can be skipped in this case upon detection of such short text, or complex sentence processing can still take place but no breakdown will take place as the text is short.

20

In another example, detection of complex sentences and simple sentences may take place first. The detected complex sentences would be processed by step 104 and the detected simple sentences would skip step 104 and be processed by step 106.

25

In traditional grammar for the English language, part of speech (POS) may refer to a category of words that have similar grammatical properties. The part of speech may explain how a word is used in a sentence. It can be said that there are eight parts of speech or eight classes in the English language: noun, pronoun, verb, adjective, adverb, preposition, conjunction, and interjection. The process of classifying words into their parts of speech and labelling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. POS based segmentation at step 106 can involve POS-tagging and word segmentation based on the tagging. Specifically, a short sentence obtained by the example of Fig. 1 is split into clauses by first

30

assigning a POS label to each word in the short sentence, and then splitting the short sentence into clauses along all conjunctions that are present in the short sentence, if they exist.

5 A customized dependency and/or constituency grammatical model may be used at step 108 for target information extraction from the clauses obtained at the CLS level. In the present example, specifically, a constituency grammar model is used at step 108. Such constituency grammar model is useful for target text information extraction and learning, and adapting to different ways of text description.

10

Generally, constituent based grammars are used to analyse and determine the constituents that a sentence is composed of. Besides determining the constituents, another important objective may be to determine the internal structure of these constituents and how they link to each other. There are usually several rules for different types of phrases based on the type of components they can contain, and parse trees can be built. A constituency-based grammar model is usually used to break sentences into various constituents. This process (i.e. breaking of sentences into various constituents) may be repeated until the level of individual tokens or words is reached.

15

Grammar types refer to a set of rules for a language. They can be used to model or represent the internal structure of sentences in terms of a hierarchically ordered structure of their constituents. For instance, each word may belong to a specific lexical category and forms the head word of different phrases. These phrases can be formed based on rules called phrase structure rules. These rules (or grammar types) can be used to model or represent the internal structure of sentences in terms of a hierarchical order.

25

Phrase structure rules form the core of the constituency grammar model used at step 108 for processing the phrases or clauses obtained at the CLS level. Such phrase structure rules govern the hierarchy and ordering of various constituents in sentences. These rules can cater to two things primarily. Firstly, these rules determine what words are used to construct the phrases or constituents and secondly, these rules determine order of the constituents. To analyse semantic phrase structure, schema patterns of the phrase structure rules can be customized. The generic representation of a phrase structure rule is $S \rightarrow AB$, which depicts that a structure S consists of constituents A and B and the ordering is A followed by B.

30

In the present example, there is present an important phrase structure rule that defines how to divide a sentence or a clause. The phrase structure rule denotes a binary division for a sentence or a clause as $S \rightarrow NP VP$, where S is the sentence or clause and it is divided into the subject, denoted by the noun phrase (NP) and the predicate, denoted by the verb phrase (VP). Additional rules can be applied to break down each of the constituents further, but the top level of the hierarchy starts with a NP and VP. The rule for representing a noun phrase is of the form $NP \rightarrow [DET][ADJ]N [PP]$, wherein the parts in square brackets [...] denote that they are optional and may or may not be present in the text inputted for processing. A noun phrase usually consists of a noun (N) as the head word and may optionally contain determinants (DET) and adjectives (ADJ) describing the noun and a prepositional phrase (PP) at the right side in the syntax tree. Consequently, a noun phrase may contain another noun phrase as a constituent of it.

The rules for representing verb phrases can be of the form $VP \rightarrow V [MD [VP][NP][PP][ADJP][ADVP]$, where the head word is usually a verb (V) or a modal (MD). A modal is itself an auxiliary verb, but a different representation is used to distinguish it from the normal verb. This is optionally followed by another verb (VP) or noun phrase (NP), prepositional phrase (PP), adjective phrase (ADJP), or adverbial phrase (ADVP). The verb phrase can be a second component when splitting a sentence using the binary division rule and the noun phrase is the first component.

The conjunctions rules are used to join clauses and phrases and form an important part of language syntax. Words, phrases and/or clauses can be combined using conjunctions. The production rule can be denoted as $S \rightarrow S \text{ conj } S \forall S \in \{S, NP, VP\}$, where two constituents can be joined by a conjunction denoted by " conj " in the rule.

After going through the various levels of extraction i.e. P, CS SS, and CLS levels, a chunk can be formed at the CK level. A chunk formed at the CK level may be a clause or phrase containing target information.

For example, the following paragraph of text is inputted at the P Level:

"If use the iron for about fifteen minutes, there is a streak of water when it produces steam. descaling according to instructions for use. 2x done. Also, in the beginning it steamed much more than the windows even fog up, now that is much less."

The abovementioned paragraph is broken down by step 102 into the following complex sentences at the CS level:

- 1) If use the iron for about fifteen minutes, there is a streak of water when it produces steam.
- 2) descaling according to instructions for use. 2x done.
- 5 3) Also, in the beginning it steamed much more than the windows even fog up, now that is much less.

The abovementioned complex sentences are broken down by step 104 into the following simple sentences at the SS level:

- 10 1) If use the iron for about fifteen minutes,
- 2) there is a streak of water when it produces steam.
- 3) descaling according to instructions for use.
- 4) 2x done.
- 5) Also, in the beginning it steamed much more than the windows
- 15 6) even fog up, now that is much less.

The abovementioned simple sentences are broken down by step 106 into clauses at the CLS level:

- 1) If use the iron for about fifteen minutes,
- 20 2) there is a streak of water
- 3) when it produces steam.
- 3) descaling according to instructions for use.
- 4) 2x done.
- 5) Also, in the beginning it steamed much more than the windows
- 25 6) even fog up,
- 7) now that is much less.

The abovementioned clauses are broken down by step 108 into the following chunks at the CK level:

- 30 1) 'a streak of water'
- 2) 'descaling according to instructions for use'
- 3) 'steamed much more than the windows'

- To facilitate extraction/determination of target information, an N-gram word mask can be embedded in the Paragraph2Chunk structure by using a regular expression for target information extraction. For example, regular expressions may include prepositions (e.g. "in," "at," "on," "of," and "to"), indefinite articles (e.g. "a" or "an") and/or definite articles (e.g. "the"). Such regular expressions can be used to identify a target word following or preceding such word in a sentence. Depending on the application/context, weightage may be given for the occurrence of certain words preceding or following regular expressions and such weightage may be considered in the extraction/determination of target information. The N-gram word mask may come in the form of chunk grammar, which is a type of regular expression containing rules that indicate what target information needs to be extracted. As an example, with reference to the python natural language tool kit, one such regular expression is given by {<DT>?<JJ>*<NN>}, where the goal is to extract a chunk that has an optional determiner (DT), followed by any number of adjectives (JJ), and finally a noun (NN).
- 15 Optionally, a MixMo (Mixing Multiple Inputs for Multiple Outputs) learning framework (Reference: "MixMo: Mixing Multiple Inputs for Multiple Outputs via Deep Subnetworks" by Alexandre Rame, Remy Sun, and Matthieu Cord, International Conference on Computer Vision (ICCV), 2021) can be used to build an adaptive model for automatic grammar model selection or construction. Such MixMo learning framework can be used to build the customized dependency and/or constituency grammatical model used at steps 104 and 108. The MixMo learning framework provides a method for automatic grammar model selection or construction so that lesser effort is needed to build the customized dependency and/or constituency grammatical models.
- 25 The chunks obtained at the CK level can be inputted to a hybrid multi-label text classification model with a pre-trained language model and character-level bag of-words (or character-level n-gram embedding) training. Such hybrid multi-label text classification model is built to realize accurate multi-label tagging "right away". The grammar model or models embedded within the Paragraph2Chunk structure help to boost the accuracy and efficiency for multi-label text classification performed by the hybrid classification model. Part of speech (POS) and dependency labels can be used in the grammar models described earlier with reference to Fig. 1 to facilitate grammar-model based target text information extraction.
- 30

Generally, a character-level bag-of-words (or character-level n-gram) model is a model subject to character-level bag of-words training that is used for extracting known characters from text.

Character-level bag-of-words is a representation of text that describes the occurrence of characters within text data inputted for processing. It may involve two things, a vocabulary of known n characters (n-gram) and a measure of the presence of the known n characters. The n characters can be an n-gram, which is a collection of n successive characters in a given text that may include words, numbers, symbols, and punctuation. An n-gram mask may be used for the detection of specific n-grams. Such model is called a character-level "bag" of words because any information about the order or structure of characters in the text to be processed is discarded. The model is only concerned with whether known n-grams occur in the text to be processed and not where they occur in the text to be processed.

Fig. 1 and Fig. 2 are largely similar. Specifically, the flowchart of Fig. 1 illustrates a method and the Fig. 2 illustrates system architecture of an apparatus or system 200. 110, 112, 114, 116, 118, and 120 of Fig. 1 focuses on steps of a workflow for multi-labels generation or prediction and Fig. 2 focuses on the modules, components or units of an apparatus or system for multi-labels generation or prediction. 110, 112, 114, 116, 118, and 120 are collectively marked out via dashed or broken lines in Fig. 1 and Fig. 2. In Fig. 2, they are parts of a hybrid multi-label text classification model for multi-label prediction 206. 102, 104, 106 and 108 are deemed as processing modules 202 in Fig. 2. 206 refers to a hybrid model for multi-label prediction, which in general, is a multi-label text classification model. P, CS, SS, CLS and CK levels are deemed as hierarchical grammatical nodes 204 in Fig. 2. 102 to 120 are regarded as method or process in Fig. 1. These steps in Fig. 1 are performed by their respective modules 202, nodes 204 and parts of a model 206 in Fig. 2.

In the example of Fig. 1 and Fig. 2, semantic similarities of the chunks formed at the CK level are measured. To enable measurement of semantic similarity between the chunks formed at the CK level and the corresponding descriptions of their corresponding tagging or labelling classes of the chunks, the chunks are represented as vectors and embedded in the same feature space. Examples of the corresponding tagging or labelling classes of the chunks may, depending on application, include grammatical classes such as noun, pronoun, verb, adjective, adverb, preposition, conjunction, and interjection. The chunks, tagging classes of the chunks, and, minimum snippets of data of interest (i.e. any extracted target information deemed useful based on the application) can be obtained using, for instance, 1) a pre-trained language model, Sentence-BERT (SBERT); and 2) Character-level n-gram embedding where a Huffman tree

based layered Softmax is used for prediction of text (or chunk or word) category or class. Accuracy is enhanced by making use of both SBERT and Character-level n-gram embedding.

- Specifically, with reference to Fig. 1, after chunks are formed at the CK level, they are subject to processing by a pre-trained Sentence-BERT (SBERT) model at a step 110, and also subject to processing by a character-level n-gram embedding model at a step 112. The character-level n-gram embedding model is part of a fast text supervised learning model that uses a Huffman tree algorithm with a Softmax function for prediction of text category or class or label. The fast text supervised learning model is based on FastText, an open-source library developed by Facebook's AI Research (FAIR) team. Generally, the pre-trained SBERT model is applied by first generating sentence embeddings for the inputted text chunks and the label descriptions, and subsequently the cosine similarities between the embeddings of the text chunks and that of the label descriptions are computed, from which the labels with the highest cosine similarities are taken for further comparison. Note that the SBERT is already trained and no training is required for the example of Fig. 1. As for the character-level n-gram embedding and Huffman tree algorithm with a Softmax function, they are implemented as part of an integrated supervised training method provided by a library (e.g. python library) for the fast text supervised learning model, and show up as parameters in the supervised training method.
- BERT language model is an open-source machine learning framework for natural language processing (NLP). BERT is designed to help computers understand the meaning of ambiguous language in text by using surrounding text to establish context. BERT stands for Bidirectional Encoder Representations from Transformers and it is based on transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection. Historically, language models could only read input text sequentially -- either left-to-right or right-to-left -- but couldn't do both at the same time. BERT is different because it is designed to read in both directions at once. The introduction of transformer models enabled this capability, which is known as bidirectionality. Using bidirectionality, BERT is pre-trained on two different but related NLP tasks: masked language modeling (MLM) and next sentence prediction (NSP). Sentence-BERT (SBERT) is a modification of the pre-trained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity. In the example of Figures 1 and 2, SBERT is used. For example, a softmax-loss approach using the 'siamese' architecture fine-tuned on the Stanford Natural Language Inference

(SNLI) and Multi-Genre NLI (MNLI) corpora may be used to train the SBERT. Although SBERT is used in the example of Fig. 1 and Fig. 2, it should be appreciated that other suitable pretrained language models may also be used.

- 5 An n-gram can be a string of n units which is part of a larger string. These units can be both characters and entire words. A word can be simply defined as a string of x characters, separated by a space on both sides. When using character n-grams for classification, spaces may be included, as word borders are also important information.
- 10 Character-level n-gram embedding involves, for instance, incorporating (or embedding) n-gram character representations during the training phase of the language model. Concretely, this means that the algorithm takes the semantic relatedness between characters (or n-grams) into account during training. The aim is that this will lead to better pattern recognition, since now it does not only look for character occurrence patterns, but also for semantical patterns. For
- 15 instance, the model or system can be trained to know that certain characters (or character n-grams) are semantically more related than other characters. It does this by means of statistical inference. For instance, certain characters (or character n-grams) will probably occur more often in some contexts than the other characters. Based on that observation, the model or system will infer that they are semantically more related. This knowledge is useful during the classification of
- 20 short text (phrases or clauses), words or sentences to the right labels. Focusing on n-grams may allow short or abbreviated forms of a word to be detected.

The softmax function described takes, for instance, as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the

25 exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval that allows the components to add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities.

30 The key reasons behind using the Huffman tree algorithm in the fast text supervised learning model is to ensure that the lookup time is faster for more frequent outputs, and to account for the fact that the label distribution may be imbalanced. FastText, the open-source library developed

by Facebook's AI Research (FAIR) team used for the fast text supervised learning model efficiently handles large amounts of text data and provides tools for:

Text Representation: FastText learns word representations (embeddings) that capture semantic meaning.

- 5 Text Classification: It builds classifiers to assign chunks (CK level) to one or multiple categories or labels.

Text Similarity Computation: It computes similarity between texts.

- 10 FastText combines some concepts including representing sentences with bag of words and bag of n-grams, as well as using sub-word information, and sharing information across classes through a hidden representation. A hierarchical softmax that takes advantage of the unbalanced distribution of the classes is used to speed up computation. These different concepts are being used for two different tasks: efficient text classification and learning word vector representations.

- 15 Generally, deep neural networks are slow to train and test, which limits their use on very large datasets. FastText helps to solve this problem. To be efficient on datasets with very large number of categories, it uses a hierarchical classifier instead of a flat structure, in which the different categories are organized in a tree (think binary tree instead of list). This reduces the time complexities of training and testing text classifiers from linear to logarithmic with respect to the
- 20 number of classes. FastText also exploits the fact that classes are imbalanced (some classes appearing more often than other) by using the Huffman algorithm to build the tree used to represent categories. The depth in the tree of very frequent categories is therefore smaller than for infrequent ones, leading to further computational efficiency. Specifically, the Huffman tree used helps to minimize expected path length from root to leaf.

25

FastText also represents a text by a low dimensional vector, which is obtained by summing vectors corresponding to the words appearing in the text. In FastText, a low dimensional vector is associated to each word of the vocabulary. This hidden representation is shared across all classifiers for different categories, allowing information about words learnt for one category to be

30 used by other categories. Such kind of representations are called bag of words and ignore word order. In FastText, vectors are used to represent word n-grams to take into account local word order, which is important for many text classification problems.

FastText is stated in the present disclosure as a preferred library for the supervised learning model to achieve the good performance showcased in the later parts of the present disclosure. However, the supervised learning model for text classification used for the example of Fig. 1 and Fig. 2 is not limited to FastText. Other suitable libraries for building the supervised learning model for text classification can also be used. For example, other libraries that may be suitable for building the supervised learning model are TensorFlow (an open-source library), Gensim (a Python library), SpaCy (another Python library), etc. Performance is still expected to be fairly good when other libraries are used because of the proposed grammar model-based analysis and the proposed competing analysis by an unsupervised learning model (e.g. SBERT) and a supervised learning model performed in the examples of the present disclosure.

In the example of Fig. 1 and Fig. 2, the probability from the root node to each leaf node would be calculated and the highest probability would be selected as the target (or predicted) text category (or class or label) and the following equation (1) can be applied:

$$p(w_i) = \prod_{j=1}^{L(w)-1} \sigma(\text{sign}(w_{i,j}) \cdot \theta_{n(w_{i,j})}^T h) \quad \text{----- (1)}$$

where
 $\theta_{n(w_{i,j})}$ represents the vector of the non-leaf node, $n(w_{i,j})$, as the output vector;
 h represents the output value of the hidden layer, which is calculated from the vector of the input word; and
 $\text{sign}(w_{i,j})$ represents a special function and its value is $\{-1, 1\}$.

With reference to Fig. 1, the SBERT model (used at step 110) and the Character-level n-gram embedding model (used at step 112) would output predicted labels for inputted chunks and probability data indicative of the probability of the predicted labels being the right labels. Given two or more sets of text and their corresponding embeddings (i.e. labels), semantic similarity of the embeddings of the two or more sets of text can be acquired by cosine similarity measurement. Accordingly, a probability comparison step 114 can be performed based on the cosine similarity measurement.

Semantic classification refers to text classification wherein predefined classes, labels or categories are assigned to text for task completion. Such semantic classification is performed at a step 116 by taking into consideration the data of the probability comparison of step 114.

- The objective of steps 110 to 120 is to obtain a set of predicted labels that best classifies a set of chunks obtained at the CK level. The set of predicted labels may be obtained by collecting labels or tags from the set of chunks where highest cosine similarity scores are generated for each of the chunks. Such cosine similarity scores are probability data and can be referred to as weightages of the labels or tags. Labels and corresponding sub-labels can be assembled based on such weightages at a step 118. Multi-label tagging or classification for the chunks will be recorded or updated based on information of the weightages of the assembled labels and/or corresponding sub-labels at a step 120.
- With regard to step 120, multi-label classification or tagging is an AI text analysis technique that automatically labels (or tags) text to classify it by topic. This differs from multi-class classification because multi-label can apply more than one classification tag to a single text. Multi-label classifiers are able to output several tags at the same time. For example, one could tag "Movies" by genre: Comedy, Drama, Romance, Horror, etc. A particular movie can be tagged as both "comedy" and "drama". In another example, multi-label classifiers can be used for customer service such as tagging a customer ticket with more than one label, so that it could be brought to the attention of multiple departments.
- A comparison on semantic sentence splitting to showcase the performance of the Paragraph2Chunk based analysis (i.e. 102 to 108 and P Level to CK Level) of the examples of Fig. 1 and Fig. 2 is provided below. Upon studying research on methods of paragraph or sentences splitting, it has been observed that these methods mostly focused on selecting specific grammatical elements or indicative lexical features to split sentence cluster according to the number of such features. In contrast, the proposed method according to the example of Fig. 1 and Fig. 2 incorporates customizable dependency/constituency grammar models on each level from paragraph to relative clauses for semantic sentence splitting, which achieves more reasonable results. The following two examples are text used to illustrate the differences between popular methods of "sentence splitter" in the state-of-the-art and the Paragraph2Chunk based analysis of the examples of Fig. 1 and Fig. 2. The text of the following two examples are text inputted to state-of-the-art apparatuses and the apparatus or system 200 of Fig. 2 for processing to interpret the meaning of the text of the following two examples. The text of the following two examples are imperfect and realistically reflect how service/system maintenance staff or users

may communicate and/or contain errors resulting from Optical Character Recognition and/or Speech-to-text recognition.

Example 1: "If use the iron for about fifteen minutes, there is a streak of water when it produces steam. descaling according to instructions for use. 2x done. Also, in the beginning it steamed much more than the windows even fog up, now that is much less."

Example 2: "#OHNE proof of purchase, please check the warranty via the serial number. R: I: Device leaves water strips when ironing. With the continuous steam burst, which is with time setting, water runs out at half the time ... T: Device is decapitated regularly. S: Workshop order."

Chunks Output (at the CK level) from Paragraph2Chunk based analysis for the abovementioned examples 1 and 2 are as follows. At this stage, multi-label classification is not performed yet.

Example 1:
Chunks: ('a streak of water'), ('descaling according to instructions for use'), ('steamed much more than the windows')

Example 2:
Chunks: ('device leaves water strips'), ('continuous steam burst'), ('is time setting'), ('water runs time'), ('device is decapitated regularly')

Chunk Outputs from three state-of-the-art methods for the examples 1 and 2 are as follows. Incorrect chunks and missed key information are stated to show poorer performance of the three state-of-the-art methods.

Method 1: "Dynamically Route Hierarchical Structure Representation to Attentive Capsule for Text Classification", by Wanshan Zheng, Zibin Zheng, Hai Wan, and Chuan Chen, In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, 2021

Example 1:
Chunks: ('use the iron'), ('about fifteen minutes'), ('streak of water'), ('produce steam'), ('iron'), ('water'), ('steam'), ('descaling'), ('steamed'), ('descaling'), ('steamed'), ('instruction for use'),
Incorrect: 'use the iron', 'about fifteen minutes', 'produce steam', 'iron', 'water', and 'steam'

Missed: 'steamed much more than windows'

Example 2:

5 Chunks: ('proof of purchase'), ('device leaves water'), ('continuous steam burst'), ('serial number'), ('check warranty'), ('water strips'), ('time setting'), ('water runs out'), ('proof'), ('warranty'), ('device'),

Incorrect: 'proof of purchase', 'serial number', 'check warranty', 'time setting', 'proof', 'warranty', and 'device'

Missed: 'device is decapitated regularly'

10

Method 2: "Open Information" Extraction from Conjunctive Sentences", by Swarnadeep Saha and Mausam, In Proceedings of the 27th International Conference on Computational Linguistics, 2018

Example 1:

15 Chunks: ('If use the iron'), ('about fifteen minutes'), ('a streak of water'), ('when it produces steam'), ('descaling'), ('according to instructions for use'), ('in the beginning'), ('it steamed much more than the windows even fog up'), ('is much less')

Incorrect: 'proof of purchase', 'serial number', 'check warranty', 'time setting', 'proof', 'warranty', 'device'

20 Missed: 'device is decapitated regularly'

Example 2:

25 Chunks: ('proof of purchase'), ('check the warranty'), ('via the serial number'), ('device leaves water strips'), ('check warranty'), ('ironing'), ('With the continuous steam burst'), ('with time setting'), ('water runs out at half the time'), ('decapitated regularly'), ('Workshop order')

Incorrect: 'proof of purchase', 'check the warranty', 'via the serial number', 'check warranty', 'ironing', 'with time setting', 'Workshop order'

30 Method 3: "Discourse-Based Sentence Splitting", by Liam Cripwell, Joel Legrand, and Claire Gardent, The 2021 Conference on Empirical Methods in Natural Language Processing, 2021

Example 1:

Chunks: ('If use the iron for about fifteen minutes'), ('there is a streak of water when it produces steam'), ('descaling according to instructions for use. 2x done'), ('in the beginning it steamed much more than the windows even fog up'), ('now that is much less')

Incorrect: 'If use the iron for about fifteen minutes', 'in the beginning it steamed much more than the windows even fog up', 'now that is much less'

Missed: 'steamed much more than the windows'

Example 2:

Chunks: ('proof of purchase'), ('please check the warranty via the serial number'), ('device leaves water strips when ironing'), ('with the continuous steam burst, which is with time setting, water runs out at half the time')

Incorrect: 'proof of purchase', 'please check the warranty via the serial number', 'with the continuous steam burst, which is with time setting, water runs out at half the time'

Missed: 'continuous steam burst', 'water runs time', 'device is decapitated regularly'

15

In the experiments conducted above for comparison, specifically, "Grammar model based semantic classification with Paragraph2Chunk structure" (i.e. only steps 102 to 108 and P to CK Levels) of Fig. 1 was tested and compared with 3 popular state-of-art methods. The data sets are real customer feedback data used for defected product analysis. The ground truth is built by manual labelling with International Repair Information System (IRIS) code. It can be seen that the chunks produced by the method 100 of Fig. 1 are more reasonable compared to those produced by the 3 state-of-art methods.

20

More details on the example illustrated via Fig. 1 and Fig. 2 are as follows.

25

With regard to the example illustrated via Fig. 1 and Fig. 2, a new Rapidly adapted text tagging (RAMT) concept is proposed. Generally, RAMT refers to the automatic labelling of a short text (i.e. a chunk) with a flexible label set A using a text mining engine, either a single label or multiple labels, where $A \subseteq B$, B is a label assembly, $\{B \mid x_1, x_2, \dots, x_n\}$, that stores all available labels for tagging. $\forall x \in B$, B and x_i are allowed to be instantly edited (edit/add/delete) and applied, thus the tagging results can be updated without tedious supervised training process.

30

To realize rapidly adapted customer feedback tagging with multiple labels (multi-label), a text mining AI engine, RAMT engine and its key modules are created for short text rapid, flexible multi-

label tagging. This RAMT engine overcomes the following drawbacks and limitations that existed in current text mining solutions: 1) complicated supervised training process, 2) limited and fixed labels (B) to be used, and 3) low accuracy and efficiency in customer feedback/short text sorting and analysis. The following sections describe details of the RAMT engine and its key modules.

5

Fig. 3 shows an apparatus based on the example of Fig. 1 and Fig. 2. Specifically, the apparatus comprises a RAMT engine 300, the key modules of the RAMT engine as well as supporting software modules that can be used, for instance, for applications such as short text/customer feedback analysis. Generally, all the raw contextual information, including text data and labels' definition, are stored in a SQL/No SQL database 302. Such raw contextual information, including text data and labels' definition are pre-determined and they depend on the application of the RAMT system or engine 300. The data from the SQL/No SQL database 302 are in the form of data streams to the RAMT engine 300 and they are transmitted via a data stream module 304, where specific format and data transformation can be defined. For instance, raw information from the SQL/No SQL database 302 may include data in different formats and they have to be transformed into a specific data format for processing by the RAMT engine 300 so as to produce consistent outputs. The data stream module 304 is a module that is able to read data from the SQL/No SQL database 302. The primary function of the data stream module 304 is to read and extract relevant columns of the raw data (containing data for providing the input text for the P Level processing in Fig. 1 and 2) from a user's input file. In addition, the data stream module 304 also provides a specific format that the data has to conform to for further data transformation.

A multi-language translation module 306 is provided alongside a text data normalization module 308, which is used for text data cleaning, expanding contractions, words correction, stemming and lemmatization, as well as tokenization. The multi-language translation module 306 is also used to translate text from one language to one or more preferred languages if required. The multi-language translation module 306 processes data from the data stream module 304.

The text normalization module 308 processes data from the data stream module 304. It normalises text data where required. The text normalization module 308 takes translated text from the multi-language translation module 306 as input, and outputs a pre-processed text, wherein this pre-processed text serves as input for the Paragraph2Chunk (paragraph-to-chunk) module 310. The text normalization module 308 can contain a few functions for text preprocessing, which includes text data cleaning, expanding contractions, word correction, stemming, lemmatization,

and/or tokenization. One or more of these functions can be performed as required. The RAMT engine 300 processes normalized text data from the text data normalization module 308. The functions of the text normalization module 308 are explained below.

5 Text Data Cleaning Function:

Purpose: Removing unwanted or irrelevant information from text data.

Tasks: Removing special characters, numbers, and punctuation. Eliminating stop words (common words like "the," "and," etc.). Ensuring consistent formatting (e.g., converting to lowercase).

10 Expanding Contractions Function:

Purpose: Expanding shortened words or phrases.

Example:

Original: "I'd love to go."

Expanded: "I would love to go."

15

Word Correction:

Purpose: Fixing typographical errors or misspelled words.

Tasks: Identifying and replacing incorrect words. Standardizing variations (e.g., "colour" to "color").

20 Stemming:

Purpose: Reducing words to their root form.

Example:

Original: "running"

Stemmed: "run"

25

Lemmatization:

Purpose: Converting words to their base or dictionary form.

Example:

Original: "better"

30 Lemmatized: "good"

Tokenization:

Purpose: Breaking text into smaller units (tokens).

Example:

Original: "Natural language processing is fascinating."

Tokens: ["Natural", "language", "processing", "is", "fascinating"]

In the present example, the key modules of the RAMT engine 300 includes a Paragraph2Chunk (paragraph-to-chunk) module 310, a pretrained language model 312 (Bidirectional encoder representations based language model) and a fast text supervised learning model 314 (e.g. a character-level bag-of-words model or character-level n-gram embedding model) providing competing analysis, and two label set processing modules 316 and 318 working together to generate a flexible label set for each data entry. The fast text supervised learning model 314 is based on FastText, an open-source library developed by Facebook's AI Research (FAIR) team. The Paragraph2Chunk module 310 may comprise a dependency grammatical model 320, a constituency grammatical model 322 and/or a rules-based model 324. The Paragraph2Chunk module 310 breaks down paragraphs (P) to chunks (CK) in the manner described with reference to Fig. 1. An example of the dependency grammatical model 320 is the model used at step 104 of Fig. 1 and an example of a constituency grammatical model 322 is the model used at step 108 of Fig. 1. The model used at step 108 of Fig. 1 can also be said to be the rules-based model 324 because as mentioned earlier, it can apply phrase structure rules, which govern the hierarchy and ordering of the various constituents in the sentences. Such phrase structure rules also define how to divide a sentence or a clause. Additional rules can also be applied to break down each constituents further e.g. break down into a noun phrase, determinants (DET), adjectives (ADJ) describing the noun and/or a prepositional phrase (PP). Furthermore, the model used at step 108 of Fig. 1 can apply rules for representing verb phrases and conjunctions rules.

The pretrained language model 312 can be a bidirectional encoder representations based language model. For example, the language model 312 can be the SBERT model used at step 110 of Fig. 1. The two label set processing modules includes a label assembly update module 316 for updating labels and a label set consolidation module 318 for consolidating labelled sets. The output of the RAMT engine 300 can be directly sent to other modules of backend 328 (e.g. for model updating, word cloud, bigram, frequency analysis, etc.), frontend 326 (e.g. user facing) and/or databases (for data storage/updating; not shown in Fig. 3). Such backend 328, frontend 326 and/or databases (not shown in Fig. 3) are configured based on the application of the RAMT engine 300. Generally, they include a feedback (backend) system for machine learning and model updating, one or more databases to store and update labels/classes and scores of labelled chunks/chunk sets, and/or a (frontend) data presentation interface (e.g. Graphical User Interface)

for presenting and/or user/machine updating of data labels/classes and scores of labelled chunks/chunk sets.

5 With reference to Fig. 3, the RAMT engine 300 integrates a hierarchical text analysis module where the pre-trained language model i.e. the language model 312 and the fast supervised learning i.e. the fast text supervised learning model 314 can be embedded for an ensemble analysis. The label list update module i.e. 316 and the label set consolidation module i.e. 318 are linked to the learning module 314 and the analysis module 312 to enable rapidly adapted multi label tagging. Specifically, the label assembly update module 316 is developed for target label
10 update in the fast text supervised learning module 314 and the bidirectional encoder representations based language model (that involves pre-trained language model based analysis) 312 respectively. The idea behind the flexible label update of module 316 is to allow users to define any labels that can be applied to the supervised/unsupervised learning model i.e. 314 or the pre-trained language model i.e. 312 immediately. In this way, the text analysis results can be
15 immediately updated by new labels or new definition, without long-term large data sets training to update new labels.

In the present example, there are two channels to update the labels and their definitions for the Label Assembly Update module 316. With regard to the first channel, a graphical user interface
20 (GUI) may be developed as the frontend 326. The frontend 326 can include software used to facilitate users or machines to define new labels, edit label names and descriptions. Specifically, the GUI is used for label assembly customization (i.e. to add/edit/delete) and it is advantageously used to provide updates for the Bidirectional encoder representations based language model 312 and the fast text supervised learning model 314. Thus, the overall label description can be
25 updated in a database for both the pre-trained language model 312 and the fast text supervised learning module 314.

The other or second channel to update label descriptions is to precisely customize the labels for the fast text supervised learning module 314 by feeding short text data with target n-gram features.
30 For instance, in the application of customer feedback analysis for a particular product, short text data with target n-gram features may refer to clauses or phrases that describe a particular product design issue. For example, a short text data may be "water leakage" and a target n-gram feature of this short text data may be a 4 characters (4-gram) feature "leak" that is contained in the short text data. The customization of the labels via the fast text supervised learning module 314

provides the fast text supervised learning module 314 with a list of texts, along with their corresponding labels, in the following format:

```
5      label__a1 text_1__  
      label__a2 text_2  
      ...  
      label__an text_n
```

10 Specifically, the target n-gram features involve the use of word n-grams as parameters for supervised training. In summary, the fast text supervised learning module 314 uses labelled data with customized labels and considers word n-grams to build a text classifier. The goal is to predict the most likely labels for the chunks.

15 For the first channel, the label assembly update module 316 retrieves pre-defined labels and user defined labels from the database that stores the data imported from a "Category" module in the backend 328. Through the GUI of the frontend 326, the RAMT engine 300 is able to receive real time new user additions of labels, customized labels or updated label assembly. The updated labels can be used by the bidirectional encoder representations based language model 312 to measure similarities between text data of normalized labels and customer feedback. In this
20 manner, classifications are easily updated to the label assembly update module 316 for use by the language model 312 without the need for tedious manual data labelling and training process for new / updated labels for the language model 312 and/or the fast text supervised learning module 314.

25 For the second channel, the label assembly update module 316 loads data sets of sample sentences and labels that users input through the GUI. The fast text supervised learning module 314 can split these sample sentences into character-level n-grams for specific character/letter/word sequence learning. That is, the data sets inputted by the user are used in the training of the fast text supervised learning module 314. Such training can take place when
30 the RAMT engine 300 is offline (not in operation) or online (in operation).

Therefore, a shallow bag-of-words learning structure is used in the RAMT engine 300. In this manner, a rapid learning model can be generated for relation mapping between target text features and their expected labels. With the first and second channels described above, the label

assembly is updated for both general and particular (general-particular) language model based multi label prediction. This is a new type of machine learning structure for rapid multi-label tagging that allows the RAMT engine 300 to rapidly adapt to real-time updated labels. In addition, the general-particular language model is updated by the two channels, thereby realizing rapid
5 classification using the pre-trained language model and provides high accurate classification. Flexible label update for rapid adapted multi-label classification is achieved.

Another key feature of the RAMT engine 300 to enable it to realize rapid multi-label tagging is in the hierarchical text data analysis structure for grammar model-based text understanding i.e. the
10 Paragraph2Chunk structure.

The GUI of the frontend 326 may further include an analysis module for Paragraph2Chunk based semantic text analysis.

15 Fig. 4 shows a hybrid multi-label text classification model of the RAMT engine 300, which is configured to predict labels for short text i.e. inputted chunks fed from the Paragraph2Chunk module 310 and compare probabilities of the predicted labels being the proper or right labels to filter out one or more predicted labels with high probabilities.

20 Specifically, to realize the proper labels to be assigned to a piece of target text data immediately, grammar model based text data analysis (via the Paragraph2Chunk module 310), pre-trained language model based analysis (via the bidirectional encoder representations module 312) and character-level n-gram embedding (via the fast text supervised learning module 314) are used. With the grammar model based text data analysis, the text data can be properly broken down
25 from paragraph (P Level) into text segment (chunks, CK Level) that contains the core target text information.

A competing analysis (or hybrid multi-label text classification) module is proposed and developed by comparing results generated by the pre-trained language model 312 with the fast text supervised learning model 314. In one example, the pre-trained language model 312 and the fast
30 text supervised learning model 314 may be used to predict labels of inputted chunks concurrently. For a given chunk, the pre-trained language model 312 outputs a plurality of predicted Labels A_1 and their corresponding probability values at a step 402. For the same chunk, the fast text supervised learning model 314 outputs a plurality of predicted Labels A_2 and their corresponding

- probability values at a step 404. Each predicted label has a probability value (or similarity value) indicative of the probability that the label is the right or correct label. From the plurality of predicted Labels A_1 and A_2 , predicted labels with highest probability (i.e. highest similarity) will be filtered at a step 406. The definition of low and high probability is pre-defined, for example, probability values
- 5 lower than 95% probability of the right label may be regarded as low. The output of step 406 will be one or more predicted Labels A 408 with highest or high probability. The similarity measurement between the candidate labels and processed chunk data may be calculated by Cosine similarity.
- 10 For instance, let c be a text chunk (CK level) with corresponding sentence embedding v_c , and let $l_1, \dots, l_n$ be the label descriptions with corresponding sentence embeddings $w_1, \dots, w_n$ respectively. The cosine similarities $s_1, \dots, s_n$ of the sentence embedding v_c and the respective sentence embeddings $w_1, \dots, w_n$ is given by their respective dot products, that is, $s_i = v_c \cdot w_i$ for all $i = 1, \dots, n$.
- 15 In another example, also making reference to Fig. 4, to make chunk classification more accurate and efficient, firstly, the chunks, CK, to be processed are assigned predicted labels A_1 (and probability values) by the pre-trained language model 312 at step 402. Secondly, the fast text supervised learning model 314 is used to examine the predicted labels A_1 having a specific range
- 20 of probability values (e.g. such range can be indicative of high or low probability) produced by the pre-trained language model 312. The fast text supervised learning model 314 conducts this examination by processing the chunks with predicted labels A_1 having the specific range of probability values instead of processing the same chunks as the pre-trained language model 312 to obtain label probability values for the chunks with predicted labels A_1 having the specific range
- 25 of probability values. The fast text supervised learning model 314 then outputs predicted labels A_2 and their corresponding probabilities (that takes into consideration the examination) at the step 404. The definition of low and high probability is pre-defined, for example, probability values lower than 95% probability of the right label may be regarded as low. The predicted labels outputted from the pre-trained language model 312 and the fast text supervised learning model 314 with
- 30 highest probability (i.e. highest similarity) will be filtered at a step 406. The output of step 406 will be one or more predicted Labels A 408 with highest or high probability. The similarity measurement between the candidate labels and processed chunk data is calculated by Cosine similarity.

For example, the precise mechanism for step 406 described in the preceding paragraph can be as follows:

Given a pre-defined probability (threshold) p , and a chunk with assigned predicted label A_1 and probability value p_1 from the pre-trained language model 312, and predicted label A_2 and probability value p_2 from the fast text supervised learning model 314, the process or filter first performs the following:

1) If both labels A_1 and A_2 are the same, the label A (final predicted label 408) is set to be equal to A_1 , and the final probability for this chunk is set to be the higher of p_1 and p_2 .

2) If label A_1 is different from label A_2 , and if p_1 is greater than p , the label A (final predicted label 408) is set to be equal to A_1 , and the final probability for this chunk is set to be equal to p_1 . Otherwise, the label A is set to be equal to A_2 , and the final probability for this chunk is set to be equal to p_2 .

Subsequently, the final labels (i.e. the Labels A 408) for all the chunks are aggregated together, with duplicate labels removed where necessary.

The RAMT engine 300 compared to an ordinary text mining engine that is widely used in current text classification or tagging is described below.

Fig. 5 illustrates the ordinary text mining engine 500. Specifically, Fig. 5 shows a widely adopted structure and modules of a text mining engine 504 used in state of-art research and commercial solutions. Inputs 502 to the text mining engine 504 are, for instance, data sources from the Internet (Social media, forums, blogs ...) and from Enterprise repositories (Enterprise information system, local file repositories ...). Data sources from the Internet may be in the form of comments, news, tweets, etc. Data sources from the Enterprise repositories may be in the form of emails, surveys, phone records, etc.

The text mining engine 504 generally involves steps of configuration, data acquisition, data pre-processing, feature extraction, feature selection, Training Model Construction (for machine learning-based approach only), Sentiment Lexicon Construction (for lexicon-based approach only) and Sentiment Classification. The outputs of the text mining engine 504 are labels/classes and scores to be transmitted to applications 506, which may include software and/or hardware, for executing application specific tasks.

The data sources for the RAMT engine 300 of Fig. 3 can be the same as the data sources described with reference to Fig. 5. However, the algorithm of the RAMT engine 300 is different from the ordinary text mining engine 500 and other conventional text mining engines at least in the following ways.

5

In terms of methodology, instead of using traditional word or sentence embedding with slide windows to measure similarity word/phrase/sentence, the RAMT engine 300 involves a grammar model based hierarchical semantic classification method and workflow. In this model, not only the occurrence of n-gram words can be considered, more semantically, dependency and constituency of the text information e.g., "noun singular/ plural /phrase + verb form + noun singular/ plural /phrase)", is also integrated to semantically identify target text snippets and calculate semantic similarities for high-dimensional text classification.

10

Hence, a hierarchical Paragraph2Chunk structure is used in the RAMT engine 300.

15

Grammar model-based target text extraction, identification and semantic multi-label classification are involved in the RAMT engine 300.

Fig. 6 shows prior art techniques for embeddings-similarity measurement based text classifications. It can be seen that typically either a supervised learning model or an unsupervised learning model that involves tedious training is used. In contrast, the RAMT engine 300 has a hybrid multi-label classification model for achieving outstanding accuracy and efficiency, based on a text supervised learning model (e.g. 314 of Fig. 3) such as a character-level bag-of-words model capable of fast learning and pre-trained language model (e.g. 312 of Fig. 3).

20

25

In terms of system architecture, comparing with the state-of-art text mining engines developed in academic research open-source packages and commercial solutions, the RAMT engine 300 provides a new engine structure and modules for multi-label tagging.

The RAMT engine 300 does not only rely on supervised learning models and the training process, but a hierarchical grammatical analysis model is considered to improve accuracy and efficiency by an competing analysis module.

30

Without manual labelling and long term training of the text data for new labels, the new labels entered or updated via the first channel or the second channel described earlier can be immediately applied to the text data. This significantly reduces computational resources and workload in maintaining the label assembly update module 316.

5

The Paragraph2Chunk based hierarchical text data analysis structure comprises grammatical models designed and developed for each level (i.e. P, CS, SS, CLS and CK).

10 A competing analysis module that integrates fast supervised learning module 314 and pre-trained language model 312 is provided. Both a supervised learning model and an unsupervised learning model are used and the best results are obtained. Accuracy is very much improved.

Label assembly update module 316 and label consolidation module 318 are provided to enable rapid adapted tagging by the competing analysis module.

15

The performance of the RAMT engine 300 compared to conventional engines is shown below. Specifically, a comparison is made between the RAMT engine 300 and frequently used text mining frameworks, SpaCy and PyTorch, Scikit learn and Keras, on accuracy and efficiency for multi label tagging.

20

SpaCy is an open-source software library for advanced natural language processing, written in the programming languages Python and Cython. SpaCy also supports deep learning workflows that allow connecting statistical models trained by popular machine learning libraries like TensorFlow, PyTorch or MXNet through its own machine learning library Thinc. In the test results
25 above, SpaCy and PyTorch are used.

Scikit-Learn, also known as sklearn is a python library to implement machine learning models and statistical modelling. Binary relevance treats each label as a separate single class classification.

30 Keras is a high-level, deep learning API developed by Google for implementing neural networks.

Test results of prediction accuracy with real industry data (involving 5 sets of test data) are provided in table 1 as follows.

Table 1:

Model	Test Data 1	Test Data 2	Test Data 3	Test Data 4	Test Data 5
SpaCy + PyTorch	83.5%	88.1%	86.3%	89.8%	82.2%
Scikit-learn (Binary relevance)	78.1%	68.6%	75.2%	88.5%	66.4%
Neural Network (Keras)	58.3%	52.7%	45.8%	68.5%	46.3%
Rapidly Adapted Multi-label Tagging (RAMT)	94.2%	96.5%	95.8%	97.2%	92.7%

Table 2 below shows comparison between the state-of-the-art text mining solutions and the RAMT engine 300 for text data multi label tagging (multi-level classification).

5

Table 2:

Solutions /functions	Key phrase extraction	Rapid label learning	Real-time label assembly update	Thematic tags	Multi-label tagging	Grammar model based analysis
Thematic platform	Present	Not Present	Not Present	Present	Present	Present
Kapiche	Present	Not Present	Not Present	Present	Not Present	Present
Expert.ai	Present	Not Present	Not Present	Present	Present	Not Present
Lexalytics	Present	Not Present	Not Present	Present	Not Present	Not Present
Amazon Comprehend	Present	Not Present	Not Present	Present	Present	Present
Google Cloud Natureal Language	Present	Not Present	Not Present	Present	Not Present	Not Present

Microsoft Azure Text Analytics	Present	Not Present	Not Present	Present	Not Present	Not Present
Rapidly Adapted Multi-label Tagging (RAMT)	Present	Present	Present	Present	Present	Present

In summary, the RAMT engine 300 comprises the following features.

- The RAMT engine 300 is a text mining engine for target text information identification and extraction that has a Paragraph2Chunk (Paragraph-Complex Sentence-Simple Sentence-Clause-Chunk) structure embedded with customizable grammar models and text processing pipeline for multi-label tagging. Specifically, the text mining engine 300 enables Grammar model-based analysis that:
- 1) allows grammar models customization,
 - 2) uses Grammar model learning with machine learning methods, and
 - 3) uses Grammar model based logic analysis (e.g. the rules proposed for constituency grammar model).
- Key phrase extraction is involved to Identify target text information in terms of syntactic and semantic rules (similarity level).
- The text mining engine 300 has a software module for flexible label assembly management that allows users to customize and integrate new labels for rapid adapted multi-label tagging. It allows users to customize text features e.g. through the first and second channels described earlier and includes an n-gram sequence for fast training. Specifically, the fast text supervised learning module 314 of the text mining engine 300 includes the word n-gram as a parameter for training, where the purpose of the word n-gram parameter is to improve the classification performance by taking word order into consideration during training. Real-time label assembly update is achieved as users are allowed to add, edit, delete and customize labels and their descriptions. Text labelling can be rapidly and adaptively updated with new labels added or modified by users. In this manner, rapid label learning can be achieved through Rapid text labelling (tagging) without tedious supervised training process.

The text mining engine 300 has a software module for standardizing multi-language text processing and multi-label tagging. Automatic multi-language detection and translation (e.g. performed via the multi-language translation module 306) is involved.

5

The text mining engine 300 involves multi-label text classification with high-dimensional label set (high dimension multi-label tagging). In other words, there is no limit to the number of labels that a user can define in a system or apparatus comprising the engine 300. Thematic tags that involve tags clustering and hierarchical regrouping are also in use. The other modules of the backend 328 (which includes word cloud, bigram and frequency analysis etc.) involves thematic tags clustering and hierarchical regrouping, so as to give the users an overview of the common themes, be it captured via the labels or through the word cloud/bigram analysis that are captured in the customer feedback.

10

15 The text mining engine 300 can be configured to use only a small proportion (such as < 5%) of data for fast training. In particular, given a dataset, for instance, a set of customer feedback data of returned and rejected products, the comments or text that are commonly mis-classified by the pretrained language model 312 can be used to construct the training data for training the fast text supervised learning model 314. The percentage of these comments that are used to train the fast
20 text supervised learning model 314 will form only a small proportion (such as < 5%) of the original set of data (i.e. the customer feedback data of returned and rejected products).

Use of the RAMT engine 300 in a specific application is described as follows.

25 There is present a real world challenge or technical problem in the manufacturing industry concerning rejected product analysis from customer feedback.

Fig. 7 shows a flowchart illustrating a customer feedback process. After a product 706 is launched in the market, a customer 708 that has in possession the product 706 may provide feedback on the product 706. Such feedback may contain the customer's comments on the product 706 to, for instance, explain why the product 706 is rejected by the customer 708. Customer feedback analysis 710 has to be conducted to understand why the product 706 is rejected. The customer's feedback may contain information pertaining to issues in product design 702, issues in the

30

manufacturing process steps 704 (i.e. step #1, step #2, ..., step #n of the manufacturing process) of the product 706.

5 A specifically designed pipeline for short text data pre-processing based on the RAMT engine 300 can be provided to analyse the customer feedback on the product 706 and categorize (or classify or label) target information in the feedback and link it to an issue in product design 702, and/or an issue in one or more steps in the manufacturing process. This can help to notify the product designer and/or the manufacturer to improve the design and/or a particular step in the manufacturing process. There are two challenges faced.

10

Challenge 1:

It is challenging to accurately understand contextual information, and measure semantical similarity of its key information to predefined labels, which can be said to be a fuzzy multi-label text classification. Customers do not provide their feedback in a structured manner and the way
15 each customer provide their feedback for the same issue can be different. Furthermore, customers may not write in perfect language and may write in a short or abbreviated manner.

For example, a customer feedback for a product may be: "If use the iron for about fifteen minutes, there is a streak of water when it produces steam. descaling according to instructions for use. 2x
20 done but problem remains. Also, in the beginning it steamed much more than the windows even fog up, now that is much less."

The preferred labels for the customer feedback above are: water leakage, descaling problem, too much steam, and unstable ironing or poor performance. Each of these labels is linked to issues
25 in product design 702 and/or one or more manufacturing steps.

Challenge 2:

It is challenging to adapt the required multi-label classification to a dynamic labelling system. Instead of using predefined labels, a dynamic label system can be used for multi-tagging.
30 However, usually there is insufficient data for new label learning, and the training process is time consuming.

In order to build an automatic and accurate text mining solution for rejected product analysis, a text processing and analysis system can be developed based on the specially designed semantic

classification engine i.e. the RAMT engine 300 of Fig. 3 and its supporting modules. This is an end-to-end solution for semantic customer feedback analysis, from text processing to multi-label tagging. Comparing with the popular state-of-the-art solutions, the RAMT engine 300 and the system (or apparatus) including it can achieve rapid and adaptive label update, and accuracy can reach 95.28% on average.

Firstly, the system can process multi-language customer feedback via the multi-language translation module 306 and semantically identify the target information, e.g., different types of product defect problems, from short-text and long-text data broken down using the Paragraph2Chunk module 310. Secondly, comparing with state-of-the-art methods, the system achieves high accuracy for high dimensional multi-label tagging, with a hierarchical, grammatical method that integrates fast supervised learning model (i.e. the fast text supervised learning module 314) and pre-trained language model (i.e. the pre-trained language model 312). Thirdly, the system adapts fast to a dynamic label system, which allows users to freely change labels and their description via the first channel and second channel described earlier. This feature overcomes the rigid and limited label tagging that is widely used in the current solutions. In addition, instead of a traditional time-consuming and laborious training process, the system can work under a resource-constraint condition, e.g., only limited training and limited testing data are available.

In order to realize accurate multi-label text classification with a dynamic label system, a hierarchical grammar model based semantic text processing and classification method is provided. This is the best method so far that can solve a fuzzy multi-label text classification problem adaptive to a dynamic label system and can achieve highest accuracy (96.08% on average) comparing with widely used state-of-the-art solutions.

Firstly, the hierarchical Paragraph2Chunk text analysis structure 310 is built to extract target text snippets from customer feedback by decomposing a paragraph into different levels of text chunks. The Paragraph2Chunk structure is scalable, customizable, and comprises hierarchical decomposing functions to breakdown paragraph, complex sentences, sentences and word snippets.

Secondly, customizable dependency and constituency grammatical models are built and embedded to levels of the Paragraph2Chunk module 310 for accurate semantic identification of

key contextual information from inputted customer feedback. Instead of co-occurrence or frequency based text analysis, the grammar models provide semantic and logical text prediction. Finally, a hybrid prediction model (i.e. hybrid multi-label text classification model) is built to achieve accurate semantic multi-label classification by integrating fast supervised learning structure that can involve character-level bag-of-words training (i.e. 314) and unsupervised label prediction with language model built by, for instance, a bidirectional learning model (i.e. 312).

Examples of the present disclosure may have the following advantages and applications.

(i) Customer feedback analysis and product improvement: Customer care representatives are always bombarded with tons of requests and queries which can become difficult to handle and lead to damage to brand reputation and drive away customers. Enterprises can automatically provide high-quality service to their customers and improve their customer care services significantly if an apparatus or system according to examples of the present disclosure is used.

(ii) Business intelligence and decision-making support: Organizations and business companies have been leveraging text mining techniques as part of their business intelligence. Apart from providing profound insights into customer behavior and trends, examples of the present disclosure can help companies to strengthen their automatic text information collection, classification and tagging, thus, giving them a competitive advantage in the market.

(iii) Digital marketing: Enterprises can run contextual web ad campaigns that bring them a high region of interest by using an apparatus or system according to examples of the present disclosure. By understanding the context on a webpage with the help of the text mining software, enterprises can place advertisements that are relevant to the information contained in the webpage.

(iv) Fraud detection: Examples of the present disclosure can provide an accurate, efficient, flexible solution to Insurance and finance companies. By combining the outcomes of text analyses with relevant structured data, these companies are able to process claims swiftly as well as to detect and prevent frauds.

(v) Social Media Analysis: Examples of the present disclosure can be integrated to track and interpret texts generated online from the news, blogs, emails, etc., allowing management to

understand the reaction of people who are interacting with a particular brand and online content. The analysis will enable their data analysis system to accurately understand 'what's hot and what's not' for the target audience.

5 The apparatus or system 200 of Fig. 2 and the RAMT engine 300 of Fig. 3 may be in the form of a computing or mobile device, for example, desktop computer, laptop computer, smart phones, tablet devices, and other handheld devices. It may have the elements of an apparatus 802 shown in Figure 8. Any image processor, controller or processor required for graphical user interfaces described in the present disclosure may also have the same elements as that described and
10 shown for the apparatus 802. The apparatus 802 is able to communicate with external devices 804 through a communication network 808, such as, a wired or wireless network (e.g. WiFi), mobile telecommunication network and the like. The apparatus 802 may be a remote server. External devices 804 may include servers that cooperate with the apparatus 802 to perform the functions of the apparatus or system 200 of Fig. 2 or the RAMT engine 300. The apparatus 802
15 may host a website (or web portal) and external devices 804 may include devices that would access the website hosted by the apparatus 802 to use the functions of the apparatus or system 200 of Fig. 2 or the RAMT engine 300. Login may be required to access the website. The external devices 804 may also include a database accessible to the apparatus 802 and which is required for the operation of the functions of the apparatus 802.

20

Specifically, the apparatus 802 may comprise a number of individual components including, but not limited to, processing unit (or processor) 816, a memory 818 (e.g. a volatile memory such as a Random Access Memory (RAM) for the loading of executable instructions 820, the executable instructions defining the functionality the apparatus 802 carries out under control of the processing
25 unit 816. The apparatus 802 also comprises a network module 825 allowing the apparatus to communicate over the communications network 808 (for example the internet). User interface 824 (internal to the apparatus 802) may be provided for user interaction and may comprise, for example, conventional computing peripheral devices such as display monitors, mouse, computer keyboards and the like. The display may show a graphical user interface such as the GUI
30 described earlier for editing, adding, deleting, updating labels. The apparatus 802 may also comprise one or more databases 826. It should also be appreciated that the one or more databases 826 may not be local to the apparatus 802. The one or more databases 826 may include a cloud database, which may comprise data to be used by the grammar model or models, the text supervised learning model, and the pre-trained language model described in the present

disclosure. The external device 804 may be one of the one or more databases 826. The SQL/NoSQL database 302 of Fig. 3 may be one of the one or more databases 826.

5 The processing unit 816 may be connected to external input/output devices such as a computer mouse, keyboard/keypad, a display, headphones or microphones, a video camera and the like (not illustrated in Figure) via Input/Output (I/O) interfaces 822. The components of the processing unit 816 typically communicate via an interconnected bus (not illustrated in Figure 8) and in a manner known to the person skilled in the relevant art.

10 The processing unit 816 may be connected to the network 808, for instance, the Internet, via a suitable transceiver device (i.e. a network interface) or a suitable wireless transceiver, to enable access to e.g. the Internet or other network systems such as a wired Local Area Network (LAN) or Wide Area Network (WAN). In addition to connecting to wired devices, the processing unit 816 of the apparatus 802 may also be connected to other wireless devices via a suitable wireless
15 transceiver device (not shown in Fig. 8) e.g. a WiFi transceiver, Bluetooth module, Mobile telecommunication transceiver suitable for Global System for Mobile Communication (GSM), 3G, 3.5G, 4G, 5G telecommunication systems, or the like.

The software and one or more computer programs installed in the apparatus 802 may include
20 one or more software applications for instant messaging platform, audio/video playback, internet accessibility, operating the apparatus 802 (i.e. operating system), network security, file accessibility, database management etc. The software and one or more computer programs may be supplied to the user of the apparatus 802 encoded on a data storage medium such as a CD-ROM, on a flash memory carrier (including USB drive) or a Hard Disk Drive, and are to be read
25 using a corresponding data storage medium drive for instance, a data storage device (not illustrated in Figure 8). Such application programs may also be downloaded from the network 808. The application programs are read and controlled in its execution by the processing unit 816. Intermediate storage of program data may be accomplished using RAM 820.

30 Furthermore, one or more of the steps of the computer programs or software may be performed in parallel rather than sequentially. One or more of the computer programs may be stored on any machine or computer readable medium that may be non-transitory in nature. The computer readable medium may include storage devices such as magnetic or optical disks, memory chips, or other storage devices suitable for interfacing with a general-purpose computer or mobile device.

The machine or computer readable medium may also include a hard-wired medium such as exemplified in the Internet system, or wireless medium such as exemplified in the Wireless LAN (WLAN) system. The computer program when loaded and executed on such a general-purpose computer effectively results in an apparatus that implements the steps of the computing methods
5 in examples herein described.

Examples of the present disclosure may have the following features. Reference letters/numerals of elements in Figures 1 to 4 are stated as examples.

- 10 A method (e.g. 100) for multi-label text classification, the method comprising:
receiving text input;
splitting the text input into one or more sentences (e.g. CS, SS);
process the one or more sentences to obtain one or more text chunks (e.g. CK) using one or more grammar models (e.g. 104, 108);
- 15 inputting one or more text chunks to a multi-label text classification model (e.g. 206) comprising a supervised learning model (e.g. 112, 314) for text classification and a pre-trained language model (e.g. 110, 312) to predict labels for the one or more text chunks;
comparing probabilities of the predicted labels to obtain one or more predicted labels having predefined probability values. Probability values cover similarity values. Values cover scores.
- 20 The method may comprise:
splitting the text input into one or more sentences based on at least punctuation and/or spacing;
processing the one or more sentences using a dependency grammar model (e.g. 104) to obtain a simplified sentence;
- 25 processing the simplified sentence to obtain one or more clauses (e.g. CLS); and
processing the one or more clauses according to a constituency grammar model (e.g. 108) to obtain the one or more text chunks.

The processing of the simplified sentence to obtain the one or more clauses may include part-of-
30 speech based segmentation.

The pre-trained language model may involve Bidirectional encoder representations.

The one or more grammar models may be built using MixMo (Mixing Multiple Inputs for Multiple Outputs) learning framework.

5 The text input may be inputted to a multi-language translator (e.g. 306) to be translated into a predefined language for splitting into the one or more sentences.

The method may comprise:

receiving user input to add or update a label, wherein data of the label is stored in a database (e.g. 826) used by the pre-trained language model for label prediction.

10

The method may comprise:

receiving user input containing datasets used for training the supervised learning model.

15 The predicted labels may be assigned for the one or more text chunks by the pre-trained language model and the supervised learning model may be used to examine the predicted labels having a specific range of probability values produced by the pre-trained language model and produce predicted labels based on the examination.

20 The similarity measurement between candidate labels considered for label prediction and the one or more text chunks may be calculated by cosine similarity.

The supervised learning model may be built based on FastText, an open-source library developed by Facebook's AI Research (FAIR) team.

25 Text in a dataset pre-determined as commonly mis-classified by the pre-trained language model may be used to construct training data for training the supervised learning model.

An apparatus (e.g. 200, 300, 802) for multi-label text classification, wherein the apparatus comprises:

30 a processor (e.g. 816) configured to execute instructions in a memory to operate the apparatus to:

receive text input;

split the text input into one or more sentences (e.g. CS, SS);

process the one or more sentences to obtain one or more text chunks (e.g. CK) using one or more grammar models (e.g. 104, 108);

input one or more text chunks to a multi-label text classification model (e.g. 206) comprising a supervised learning model (e.g. 112, 314) for text classification and a pre-trained language model (e.g. 110, 312) to predict labels for the one or more text chunks;
compare probabilities of the predicted labels to obtain one or more predicted labels having
5 predefined probability values.

The apparatus may be operable to:
split the text input into one or more sentences based on at least punctuation and/or spacing;
process the one or more sentences using a dependency grammar model to obtain a simplified
10 sentence;
process the simplified sentence to obtain one or more clauses (e.g. CLS); and
process the one or more clauses according to a constituency grammar model to obtain the one
or more text chunks.

15 With regard to the apparatus, part-of-speech based segmentation may be used to process the
simplified sentence to obtain the one or more clauses.

With regard to the apparatus, the pre-trained language model may involve Bidirectional encoder
representations.

20 With regard to the apparatus, the one or more grammar models may be built using MixMo (Mixing
Multiple Inputs for Multiple Outputs) learning framework.

With regard to the apparatus, the text input may be inputted to a multi-language translator (e.g.
25 306) to be translated into a predefined language for splitting into the one or more sentences.

The apparatus may be operable to:
receive user input to add or update a label (e.g. via a label assembly update module 316), wherein
data of the label is stored in a database (e.g. 826) used by the pre-trained language model for
30 label prediction.

The apparatus may be operable to:
receive user input containing datasets used for training the supervised learning model.

With regard to the apparatus, the predicted labels may be assigned for the one or more text chunks by the pre-trained language model and the supervised learning model may be used to examine the predicted labels having a specific range of probability values produced by the pre-trained language model and produce predicted labels based on the examination.

5

With regard to the apparatus, the similarity measurement between candidate labels considered for label prediction and the one or more text chunks may be calculated by cosine similarity.

With regard to the apparatus, the supervised learning model may be built based on FastText, an open-source library developed by Facebook's AI Research (FAIR) team.

10

With regard to the apparatus, text in a dataset pre-determined as commonly mis-classified by the pre-trained language model may be used to construct training data for training the supervised learning model.

15

The text input for the method and apparatus may comprise human feedback, commentary or request on a matter. Such matter depends on the application. Examples of some applications are Customer feedback analysis and product improvement, Business intelligence and decision-making support, Digital Marketing, Fraud detection, and Social Media Analysis described earlier.

20

In the present disclosure, unless the context clearly indicates otherwise, the term "comprising" has the non-exclusive meaning of the word, in the sense of "including at least" rather than the exclusive meaning in the sense of "consisting only of". The same applies with corresponding grammatical changes to other forms of the word such as "comprise", "comprises" and so on.

25

While the invention has been described in the present disclosure in connection with a number of examples, embodiments and implementations, the invention is not so limited but covers various obvious modifications and equivalent arrangements, which fall within the purview of the appended claims. Although features of the invention are expressed in certain combinations among the claims, it is contemplated that these features can be arranged in any combination and order.

30

CLAIMS

1. A method for multi-label text classification, the method comprising:
receiving text input;
5 splitting the text input into one or more sentences;
 process the one or more sentences to obtain one or more text chunks using one or more grammar
 models;
 inputting one or more text chunks to a multi-label text classification model comprising a supervised
 learning model for text classification and a pre-trained language model to predict labels for the
10 one or more text chunks;
 comparing probabilities of the predicted labels to obtain one or more predicted labels having
 predefined probability values.
2. The method of claim 1, wherein the method comprising:
15 splitting the text input into one or more sentences based on at least punctuation and/or spacing;
 processing the one or more sentences using a dependency grammar model to obtain a simplified
 sentence;
 processing the simplified sentence to obtain one or more clauses; and
 processing the one or more clauses according to a constituency grammar model to obtain the
20 one or more text chunks.
3. The method of claim 2, wherein the processing of the simplified sentence to obtain the one or
 more clauses includes part-of-speech based segmentation.
- 25 4. The method of any one of the preceding claims, wherein the pre-trained language model
 involves Bidirectional encoder representations.
5. The method of any one of the preceding claims, wherein the one or more grammar models is
 built using MixMo (Mixing Multiple Inputs for Multiple Outputs) learning framework.
30 6. The method of any one of the preceding claims, wherein the text input is inputted to a multi-
 language translator to be translated into a predefined language for splitting into the one or more
 sentences.

7. The method of any one of the preceding claims, the method comprising:
receiving user input to add or update a label, wherein data of the label is stored in a database
used by the pre-trained language model for label prediction.
- 5 8. The method of any one of the preceding claims, the method comprising:
receiving user input containing datasets used for training the text based supervised learning
model.
- 10 9. The method of any one of the preceding claims, wherein the predicted labels are assigned for
the one or more text chunks by the pre-trained language model and the text based supervised
learning model is used to examine the predicted labels having a specific range of probability
values produced by the pre-trained language model and produce predicted labels based on the
examination.
- 15 10. The method of any one of the preceding claims, wherein similarity measurement between
candidate labels considered for label prediction and the one or more text chunks is calculated by
cosine similarity.
- 20 11. The method of any one of the preceding claims, wherein the text based supervised learning
model is built based on FastText, an open-source library developed by Facebook's AI Research
(FAIR) team.
- 25 12. The method of any one of the preceding claims, wherein text in a dataset pre-determined as
commonly mis-classified by the pre-trained language model is used to construct training data for
training the text based supervised learning model.
- 30 13. An apparatus for multi-label text classification, wherein the apparatus comprises:
a processor configured to execute instructions in a memory to operate the apparatus to:
receive text input;
split the text input into one or more sentences;
process the one or more sentences to obtain one or more text chunks using one or more grammar
models;

input one or more text chunks to a multi-label text classification model comprising a supervised learning model for text classification and a pre-trained language model to predict labels for the one or more text chunks;
compare probabilities of the predicted labels to obtain one or more predicted labels having
5 predefined probability values.

14. The apparatus of claim 13, wherein the apparatus is operable to:
split the text input into one or more sentences based on at least punctuation and/or spacing;
process the one or more sentences using a dependency grammar model to obtain a simplified
10 sentence;
process the simplified sentence to obtain one or more clauses; and
process the one or more clauses according to a constituency grammar model to obtain the one or more text chunks.

15. The apparatus of claim 14, wherein part-of-speech based segmentation is used to process the simplified sentence to obtain the one or more clauses.

16. The apparatus of any one of the claims 13 to 15, wherein the pre-trained language model involves Bidirectional encoder representations.

17. The apparatus of any one of the claims 13 to 16, wherein the one or more grammar models is built using MixMo (Mixing Multiple Inputs for Multiple Outputs) learning framework.

18. The apparatus of any one of the claims 13 to 17, wherein the text input is inputted to a multi-
25 language translator to be translated into a predefined language for splitting into the one or more sentences.

19. The apparatus of any one of the claims 13 to 18, the apparatus is operable to:
receive user input to add or update a label, wherein data of the label is stored in a database used
30 by the pre-trained language model for label prediction.

20. The apparatus of any one of the claims 13 to 19, the apparatus is operable to:
receive user input containing datasets used for training the supervised learning model.

21. The apparatus of any one of the claims 13 to 20, wherein the predicted labels are assigned for the one or more text chunks by the pre-trained language model and the supervised learning model is used to examine the predicted labels having a specific range of probability values produced by the pre-trained language model and produce predicted labels based on the examination.

22. The apparatus of any one of the claims 13 to 21, wherein similarity measurement between candidate labels considered for label prediction and the one or more text chunks is calculated by cosine similarity.

23. The apparatus of any one of the claims 13 to 22, wherein the supervised learning model is built based on FastText, an open-source library developed by Facebook's AI Research (FAIR) team.

24. The apparatus of any one of the claims 13 to 23, wherein text in a dataset pre-determined as commonly mis-classified by the pre-trained language model is used to construct training data for training the supervised learning model.