7-1-2013

Ontology-based annotation using naive Bayes and decision trees

Jiawei Xu Jr

Jiawei Xu

Candidate

Computer Science

Department

This thesis is approved, and it is acceptable in quality and form for publication:

Approved by the Thesis Committee:

George F. Luger , Chairperson

Jessica A. Turner

Lance R. Williams

Ontology-based Annotation using Naive Bayes and Decision Trees

by

Jiawei Xu

B. S., Business Administration, Tongji University
M. S., Micro-electronics, Fudan University

THESIS

Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science
Computer Science

The University of New Mexico
Albuquerque, New Mexico

July, 2013

ACKNOWLEDGEMENTS

This thesis is based on a collaborative research project on Text Data mining and Machine Learning between the computer science department of the University of New Mexico and Mind Research Network (MRN) in Albuquerque, USA, supported by The National Institute of Health.

I would like to thank my research advisor, Professor George F. Luger, for his support, instructions and lots of inspirations. From taking his class and doing research under his supervision, I have been introduced into the world of artificial intelligence where I found my real interest, passion, motivation.

Thanks also to my colleagues Chayan Chakrabarti and Thomas B. Jones, for working together with me on the project. Finally, lots of thanks to Dr. Jessica Turner and Dr. Matthew Turner from MRN, who have provided expert knowledge for us in the biomedical field.

Although this Master dissertation reflects a group effort to solve an interesting but difficult problem, my own contributions include:

1. Articulating the problem and summarizing the components related to the project (chapter 1);
2. Collecting and describing possible approaches to solve the problem (chapter 2);
3. Taking part in the implementation of our approach, in particular the component of f-score evaluation (chapter 3);
4. Evaluating the performance and exploring possible alternatives, in particular proposing an 'enhanced' version of the algorithm which achieves better performance (chapter 4).

ONTOLOGY-BASED ANNOTATION USING NAIVE BAYES AND DECISION TREES

by

Jiawei Xu

B.S., Business Administration, Tongji University, 2005
M.S., Micro-electronics, Fudan University, 2009
M.S., Computer Science, University of New Mexico, 2013

Abstract

The Cognitive Paradigm Ontology (CogPO) defines an ontological relationship between academic terms and experiments in the field of neuroscience. BrainMap (www.brainmap.org) is a database of literature describing these experiments, which are annotated by human experts based on the ontological framework defined in CogPO. We present a stochastic approach to automate this process. We begin with a gold standard corpus of abstracts annotated by experts, and model the annotations with a group of naive Bayes classifiers, then explore the inherent relationship among different

components defined by the ontology using a probabilistic decision tree model. Our solution outperforms conventional text mining approaches by taking advantage of an ontology.

We consider five essential ontological components (Stimulus Modality, Stimulus Type, Response Modality, Response Type, and Instructions) in CogPO, evaluate the probability of successfully categorizing a research paper on each component by training a basic multi-label naive Bayes classifier with a set of examples taken from the BrainMap database which are already manually annotated by human experts. According to the performance of the classifiers we create a decision tree to label the components sequentially on different levels. Each node of the decision tree is associated with a naive Bayes classifier built in different subspaces of the input universe. We first make decisions on those components whose labels are comparatively easy to predict, and then use these predetermined conditions to narrow down the input space along all tree paths, therefore boosting the performance of the naive Bayes classification upon components whose labels are difficult to predict. For annotating a new instance, we use the classifiers associated with the nodes to find labels for each component, starting from the root and then tracking down the tree perhaps on multiple paths. The annotation is completed when the bottom level is reached, where all labels produced along the paths are collected.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

With the rapid advance of biomedical research the amount of biomedical literature has been growing fast in recent years. For a researcher to efficiently find relevant literature that he/she is interested in is a very important task. Nowadays the main biomedical literature databases have grown into such big size that the number of abstracts they reference could go beyond millions (For instance, PubMed currently comprises over 22,000,000 abstracts[1]). Because of the enormous size, accurate and complete information is missed more often than not with common approaches which are usually based on plain text search of the literature[2].

1.1. Information retrieval

This task of finding the exact desired literature has everything to do with information retrieval. Information retrieval refers to the process of extracting information pertaining to a specific need from a set of information entities. The search targets that an information retrieval system is based on can be certain data, or metadata which is data about the collections of data, such as documents. books, journals, videos, photographs or Web pages[3]. The most common information retrieval systems are the web search engines.

In order for an information retrieval system to address information needs, they are represented by queries, which are usually in the form of formal logic statements. An information retrieval process starts with feeding a query specified

by a user into an information retrieval system. The information retrieval system is usually attached to a database system, whose data or metadata is searched according to the input query. An information retrieval system usually does not uniquely identify one entity in the database system; as in most cases there are multiple entities which may be considered related to the query, with different degrees of relevancy. The system then uses certain approaches to compute how well the relevant entities match the query, and returns the most related entities to the user.

In reality, as this information retrieval process is performed automatically by a system, the results may not always be accurate. Hence there ought to be measures to evaluate the performance of the an information retrieval system. Many different measures have been proposed and most of them assume a premise exists that every returned entity can be classified as either relevant or irrelevant to a particular query.

1.2. F-measure

One popular way to measure the performance of an information retrieval system is called *F-measure*, or *F-core*[4], which is also the approach we adopt to evaluate our work. F-measure is the "weighted harmonic mean of *precision* and *recall*"[5].

Precision is the percentage of returned entities which are considered relevant to a particular query, while *recall* is the percentage of relevant entities out of all the entities that have been returned. For example, an information retrieval system fulfills a query of identifying all men from a collection of 42 human facial images,

among which 39 are male. It returns 33 images, in which 14 turn out to be women. Therefore, the precision of this information retrieval system with respect to the query of identifying all men is 19/33, and the recall is 19/39.

The *F-measure* is computed on basis of precision and recall as follows:

$$F = 2\frac{precision \cdot recall}{precision + recall}$$

It can be also called F_1 measure, or F_1 score, in which precision and recall have the same weight. It can be generalized to F_β measure, where β is a non-negative real value, which indicates recall is β times as important as precision:

$$F_\beta = (1+\beta^2) \cdot \frac{precision \cdot recall}{\beta^2 \cdot precision + recall}$$

The most commonly used F measures are F_1 measure, F_2 measure and $F_{0.5}$ measure.

1.3. Text mining

Information retrieval involves obtaining information from all kinds of media, while the task of finding relevant literature only targets text documents. Therefore solving this problem is also related to text mining.

Text mining is a special type of data mining, whose objective is to extract desired information from text data[6]. The information concerned here usually refers to patterns or trends in the text, hence the way text mining works has

everything to do with pattern recognition[7]. A typical framework of text mining usually consists of parsing the input text, reorganizing the text in a structured manner, indentifying patterns from the structured text, and finally returning the interpretation of the text. The paring and reorganization of the text are also referred to as preprocessing, during which addition and removal of certain content are often performed.

Apart from information retrieval, text mining is also related to link and association analysis, lexical analysis such as studying frequency or distributions of words, text tagging and annotation. Typical text mining applications are text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling.

Biomedical text mining is a subcategory of text mining, which is dedicated to texts and literature in the biomedical domain. It is also considered as an interdisciplinary research field of natural language processing, bioinformatics and computational linguistics. As stated before, the electronic publications in major biomedical databases such as PubMed are growing rapidly, hence the information retrieval techniques dedicated specifically to biomedical literature have called upon more and more research interest in recent years.

1.4. Brain mapping and neuroimaging

Biomedical science is a broad category. Neuroscience is a subfield of biomedical science, which involves approaches to study the nervous system. Neuroimaging

is about "the use of different techniques to either directly or indirectly image the structure and function of the brain"[9]. Brain mapping is "a set of neuroscience techniques predicated on the mapping of biological quantities or properties onto spatial representations of the human or non-human brain resulting in maps"[10]. The idea behind this is that the normal flow of electrical impulses in brain tissue can be disrupted by injuries and diseases, such as a physical injury (e.g., concussion), toxic injury, seizure disorder, Alzheimer's disease, anoxia and brain infection (e.g., chronic Lyme encephalitis). Even common emotions such as anxiety and depression can alter brainwave activities, leaving distinct brainwave "signatures". Brain mapping is a quantitative recording of such activities. It is essentially a comprehensive analysis of brainwave frequency bandwidths on which topographic color-coded maps that show brainwave activities can be created. Brain mapping can be conceived as a higher form of neoroimaging, producing brain images supplemented by the result of additional (imaging or non-imaging) data processing or analysis, such as maps projecting behavior onto brain regions. On the other hand, all neuroimaging can be considered part of brain mapping. Functional and structural neoroimaging are at the core of the mapping aspect of brain mapping.

Cognitive neuroscience is a discipline based on experiments, whose goal is to associate structure to corresponding function with applications of psychology and neuroscience. Cognitive neoroimaging and brain mapping methods are powerful research tools for neuroscience, which have led to the generation of enormous amount of data. Given the vast amounts of published results in this field, neuroimage scientists have become increasingly interested in function-location meta-analysis, in which they pool similar studies to identify the most consistent

brain-activation patterns observed under similar experimental conditions. Meta-analytic tools that synthesize, organize, and interpret distinct segments of the cognitive neuroimaging literature have been facilitated by the Brainmap Project, a public repository of neuroimaging findings[11]. Its contributions have resulted in what is now a relatively automated pipeline from study selection to meta-analytic image interpretation. The ability to perform meta-analysis to identify replicated results is part of the toolset needed to explore the different cognitive constructs underlying similarities and differences in brain function in related disorders, such as the constellation of schizophrenia, bipolar disorder, depression and autism.

However, the ability to perform meta-analysis across experimental domains is challenged by identification of the appropriate literature. Currently, researchers manually carry out multiple searches in the PubMed database with different keywords from alternate terminologies to attempt to capture the entirety of the studies they seek. This approach is inefficient and ineffective.

The fact that relevant publications are easily missed is largely due to the widely used alternate and even competitive terminologies among neuroimaging and brain mapping publications. While the experimental psychology and cognitive neuroscience literature may refer to a certain behavioral paradigm by name (e.g., the Stroop paradigm or the Sternberg paradigm) or by function (e.g., a working memory task or a visual attention task), these paradigms can vary tremendously in the stimuli presented to the subject, and the instructions given to the subject. For example, a general task could be given totally different names such as "Sternberg Task", "Delayed Match to Sample Task", "Serial Item Recognition

Task", and "Working Memory Tasks" in different experiments.

1.5. Ontology and CogPO (Cognitive Paradigm Ontology)

The content of most brain mapping publications is about certain experiments whose results lead to certain facts about brain activity. Therefore the structure of these publications tends to follow some particular patterns. This distinct characteristic can be taken advantage of to aid the task of automatic extracting and organizing essential information from these publications.

In computer and information science, an ontology formally "represents knowledge as a set of concepts within a domain, and the relationships between pairs of concepts"[12]. It can be used to model a domain with the definition of entities and concepts together with their properties and relations by means of shared vocabulary and taxonomy. An ontology is "a structural framework for organizing information". Applications of ontology can be found in artificial intelligence, semantic web, biomedical informatics, knowledge representation, and so forth.

The Cognitive Paradigm Ontology (CogPO)[13] was created in 2009 to address the non-standard vocabulary that exists for describing behavioral tasks or paradigms in brain mapping experiments. The design of CogPO is focused on "what can be observed directly: categorization of each paradigm in terms of 1) the stimulus presented to the subjects; 2) the requested instructions; 3) the returned response". Since all paradigms consist exactly of these three orthogonal components, forming an ontology to describe paradigms becomes a "clear and

direct" approach. CogPO seeks to "represent stimuli, responses and instructions that define the conditions of the experiment in a standard format, with well-defined terms and relationships between them". The driving force behind CogPO's design is to support published experiments implementing similar behavioral task characteristics to be linked, despite the use of alternate vocabularies.

CogPO actually transfers the task of identifying the paradigm names in a plain scope into identifying a common set of hierarchical characteristics of the experiments which captures the nature of the discoveries published more accurately and avoids any ambiguity. This naturally leads to the question of how to capture the ontology terms from free text that characterizes the experimental tasks.

The National Center for Biomedical Ontology (NCBO) provides "online tools and a web portal enabling biomedical researchers to access, review, and integrate disparate ontological resources in all aspects of biomedical investigation and clinical practice to support their knowledge-intensive work"[14]. A major focus of it involves "the use of biomedical ontologies to aid in the management and analysis of data derived from those complex experiments". In order to achieve this, NCBO has developed the "NCBO annotator" as a tool for automated identification of existing ontological terms from literature text.

There is one more gap to fill. NCBO is based on the whole collection of biomedical science publications and there are too many ontological terms that are related. The terms annotated by the "NCBO annotator" can come from any ontology. They need to be further filtered and organized in order to be associated

and used by CogPO. To achieve this, a new computational resource, called "BrainMap Tracker", which integrates the NCBO's ontology annotation tools and CogPO is created to address the task of automatic annotation and identification of candidate studies for neuroimaging meta-analysis using PubMed[1].

1.6. BrainMap Tracker

This section is adapted mainly from the research proposal written by Dr. Jessica Turner for the Brainmap Tracker project (NIH R56MH097870).

The problem "BrainMap Tracker" attempts to address is the ability to rapidly identify what paradigms have been utilized to study brain activations across neuropsychiatric disorders. The work of this thesis is to focus on portions of the software methodology to realize such a tool.

To carry out this goal we begin with a set of manually curated studies archived in the BrainMap database that focus on four exemplar mental disorders: schizophrenia, autism, bipolar disorder, and depression (see **Appendix A**). These manually annotated studies offer a baseline or "gold standard" for comparison and validation with the automatic annotation algorithms we develop, as the foundation of our work.

To formalize our goals, we aim to: 1) Develop automatic annotation algorithms to extend the functionality of the NCBO Annotator, which is an annotation tool provided by NCBO; 2) Develop a search and retrieval tool for cognitive neuroimaging studies; 3) Evaluate BrainMap Tracker to identify patterns of

overlapping studies among schizophrenia, autism, bipolar disorder, and depression.

The BrainMap Tracker project will integrate CogPO's domain-specific knowledge representation capabilities and the Brain-Map database resource with the annotation capabilities of NCBO Annotator. The approach we present to achieve this objective is to use a stochastic framework to automate this integration process. We use a hierarchical version of a naive Bayes classifier, and then leverage the inherent structural relationships among the different concepts as defined by the ontology using a probabilistic decision tree model[55].

Chapter 2

Background

2.1. Machine learning

The key idea of CogPO is to characterize all the behavioral experiments by a certain set of ontological terms which are hierarchically related to each other in the representation of cognitive experiments. These terms fall into components or dimensionst, which are Stimulus Modality, Stimulus Type, Response Modality, Response Type and Instruction, which describe five aspects of the experiment accordingly. For a specific behavioral experiment, each of these five aspects could be summarized by one or more ontological terms from a limited, disjoint vocabulary, depending on the content of the experiment. For example, Stimulus Modality could be "Auditory", "Visual", "Tactile", and etc; Stimulus Type could be "Faces", "Food", "Heat", and etc. Each experiment has some tags from all these five aspects, even if the tag might be "No Stimulus", for example. Therefore, each experiment is associated with a set of five ontological terms. The internal relationship between the literature document which describes the experiment could be learned by machine learning approaches with a group of examples for automatic reasoning.

Machine learning is a subcategory of artificial intelligence about "the construction and study of systems that can 'learn' from data"[15]. All machine learning processes consist of two basic phases: learning and generalization. The learning phase refers to identifying rules and trends from a set of examples, while

generalization refers to the ability of the system to make accurate predictions on unforeseen examples according to the previous learned rules or trends. In another word, the key idea of machine learning is to extract some generally unknown probability distribution from existing data so that it can be used to produce accurate predictions on new data. Machine learning has everything to do with data mining as many techniques are used for both tasks. However, the objective is not the same. Machine learning aims at making accurate predictions by learning properties from data, while data mining emphasizes on discovering unknown properties from the data.

There are two main categories of machine learning: supervised learning[16] and unsupervised learning[17]. Supervised learning refers to "inferring a function from labeled training data". The training data is composed of a set of training examples. In supervised learning each example consists of two parts: an input vector and a corresponding output value. The training data is first parsed by certain supervised learning algorithms. Then a function between the input and output is proposed, which can be seen as a classifier or a regression function, depending on whether the output value is discrete or continuous. The proposed function is supposed to predict the correct output value when any new input vector is accepted. Unsupervised learning, however, deals with unlabeled data, which means the input vector has no desired output value. It attempts to "find hidden structure" in the data itself. In our case, as explained before, we have a desired output value, and the output is the set of CogPO components, which are five distinct ontological terms. Therefore, we will focus on the approaches of supervised learning for discrete output.

2.2. Multi-class multi-label classification

For supervised learning, when the output is of discrete value, the task of machine learning is also called classification[18]. As the output is discrete, it can be regarded as a group of categories, thus the problem can be regarded as categorizing a set of input values into these categories, or classes. An algorithm that implements classification is called a classifier.

The training data is composed of a group of examples (or instances), each of which has input and output values. The input values are also called features. As this name suggests, features are actually a vector of characteristics which describe the example instance. There are a variety of features types: binary, nominal, ordinal, numeric, and so forth. The size of features is predetermined. The output values, or the categories to classify the examples into, are also called labels. The size of labels are also prefixed. The least size of labels is two, which means a single instance described by a vector of features belongs to either one of the two classes. Such classification problems are called binary classification. On the other hand, if the size of labels are more than two, it is called multi-class classification. If each vector of features can only correspond to one label, the classification problem is single-label classification. As the majority of problems are of this type, classification problems are referred to as single-labeled by default. There are cases that each vector of features could correspond to one or more labels. Such problems are called multi-label classification, to distinguish from single-label classification problems. The approaches to solve these problems are also different correspondingly.

Our problem is by nature a text classification problem, a problem of assigning a text document into one or more topics or categories[19]. Suppose we have a brain mapping publication, we need to determine what behavioral experiment it describes, by extracting the five ontological terms which correspond to five CogPO components (Stimulus Type, Stimulus Modality, Response Type, Response Modality and Instructions) from the content of document. A full list of all possible terms for each of these five components can be found in **Appendix B**. One component can actually have multiple terms combined together to describe an behavioral experiment. For example, from our manually annotated examples, one publication with PubMed ID 30376 has two terms "Letters" and "Words" annotated for component "Stimulus Type". Since all components have more than two possibilities for ontological terms, it is a multi-label, multi-class text classification problem for each component. Therefore, our problem can be seen as five separate sub-problems of multi-class, multi-label classification.

Here we give a formal definition of a multi-class, multi-label classification problem: let X be the instance universe, and consider a set of labels $Y = \{1, ..., k\}$. The goal is to find a hypothesis $h : X \to 2^Y$ with error as low as possible, based on a set of examples $S = \{(x_i, Y_i) \mid x_i \in X, Y_i \subseteq Y, 1 \le i \le m\}$.

2.2.1. Problem transformation method

In comparison with multi-label classification, single-label classification has been well studied. SVM[20] and Naive Bayes[21] are popular single-label classifiers. It is possible to transfer a multi-label classification problem into a single-label

classification problem, then such existing single-label classifiers can be applied directly to address multi-label classification.

There are several ways to fulfill the task of transformation[22][23]. To illustrate these ideas, an example of training data set is set up as Table 1:

Instances	Features	Label set
1	X_1	$\{\lambda_2, \lambda_4\}$
2	X_2	$\{\lambda_3\}$
3	X_3	$\{\lambda_1, \lambda_2, \lambda_4, \lambda_5\}$

Table 1 Example of multi-label classification

There are four instances in the data set, which correspond to four feature sets represented by X_i. The form, size and attributes of features are not explicitly given because they do not really matter to the problem of transformation. In single-label cases, there is only one label λ_i that corresponds to each instance, while in this example three of four instances have more than one labels. A number of transformation approaches are very simple: *select-max, select-min, select-random, ignore*. The simplest of them is *ignore*, which discards all the instances with multiple labels from the training data set. The other three all transfer each multi-label set into single-label set by selecting one label out of the set and discarding the rest. To achieve this, a single-label classifier that outputs probability distributions over all classes can used to learn a ranking. The class with the highest probability will be ranked first, the class with second highest probability will be ranked second, and so forth. For the label set of each instance,

select-max simply picks the most frequent label, while *select-min* picks the least frequent label; and *select-random* picks one randomly. The outcome after transformation is shown in Table 2, from (a) to (d). It is obvious that all of these approaches discard significant amounts of information during the transformation process which is crucial to understanding properties of the data set.

Idx	Label
3	λ_3

(a)

Idx	Label
1	λ_4
2	λ_3
3	λ_4

(b)

Idx	Label
1	λ_2
2	λ_3
3	λ_1

(c)

Idx	Label
1	λ_4
2	λ_3
3	λ_1

(d)

Idx	Label
1a	λ_2
1b	λ_4
2	λ_3
3a	λ_1
3b	λ_2
3c	λ_4
3d	λ_5

(e)

Idx	Label	Weight
1a	λ_2	0.50
1b	λ_4	0.50
2	λ_3	1.00
3a	λ_1	0.25
3b	λ_2	0.25
3c	λ_4	0.25
3d	λ_5	0.25

(f)

Table 2 Simple transformation methods

More advanced approaches try to avoid that. For example, *copy* method splits an instance with multiple labels into several, each of which is distributed one label from the original label set. *Copy-weight* method further associates each of the sub-instances with a normalized probability which is dependent on the original size of the label set. The outcome of these two approaches are shown in Table 2 (e) and (f). There is still information loss with these methods because the fact that a particular instance is labeled as A and B is by nature different from the fact that it

is sometimes labeled as A and sometimes labeled as B.

Actually, a label set can also be considered as a special type of single label. Such a single label represents the specific combination of the exact labels contained in the label set. This transformation approach is called label power set (LP). If there are *n* possible labels, then there will be 2^n possibilities of label combination in total.

Idx	Label
1	$\lambda_{2,4}$
2	λ_3
3	$\lambda_{1,2,4,5}$

(a)

c	$p(c\|\mathbf{x})$	λ_1	λ_2	λ_3	λ_4	λ_5
$\lambda_{2,4}$	0.4	0	1	0	1	0
λ_3	0.2	0	0	1	0	0
$\lambda_{1,2,4,5}$	0.4	1	1	0	1	1
	$\sum_c p(c\|\mathbf{x})\lambda_j$	0.4	0.8	0.2	0.8	0.4

(b)

Table 3 Transformation by label power set method

Table 3 (a) shows the training data set after transformation; Table 3 (b) shows an example of possible probability distribution produced by LP, based on the training data set, given a new instance. The label ranking for each label is the sum of the probabilities among all possibilities. Although the computational complexity is upper bounded by $\min(n, 2^k)$, where *n* is the total number of data instances, and *k* is the total number of labels in the training data before transformation, usually the actually complexity is much smaller than 2^k. One problem of this approach is while there are 2^k labels (after transformation), the majority of them are not likely to be seen in the training data instances. This leads to a large number of labels associated with only a small number of data instances that would cause extreme

label imbalance for learning. A *Pruned Problem Transformation*[24] method has been proposed to address this problem by pruning away the label sets that occur less than a user-defined threshold and replacing them by introducing disjoint subsets of these label sets that show up more frequently in the training data instances.

Another most well-known transformation method is *Binary Relevance*(BR)[25]. It breaks the whole data set into *L* single-label subsets, where *L* is the size of the original label set. Each of these subsets focuses on one label, say λ_i. If λ_i is in the label set for one instance, this instance is labeled λ_i, or $\neg\lambda_i$ otherwise.

The data set after transformation is shown in Table.4. Since now in every subset, each instance is associated with one label, therefore it is easy to train the subsets with a *binary classifier*[20]. The problem with this approach is by treating each label separately it assumes by default that all the labels are independent of each other, while this might not be the case in many multi-label applications.

There is another advanced method of transformation by means of *label ranking*[26], which is a preference learning scenario. The original *label ranking* problem is slightly different from classification, whose goal is to predict the preference order of a set of labels when given a new instance after learning a group of training examples. An example is shown below in Table 5.

Again the details of the feature set is ignored and is represented by a person's name. It can contain any kinds of attributes such as height, weight, hobby and so forth. The second to fourth columns represent the first three German automobile

brands that person prefers. According to these information, given a new person Matt's attributes, the task is to predict his preference.

Idx	Label
1	$\neg\lambda_1$
2	$\neg\lambda_1$
3	λ_1

(a)

Idx	Label
1	λ_2
2	$\neg\lambda_2$
3	λ_2

(b)

Idx	Label
1	$\neg\lambda_3$
2	λ_3
3	$\neg\lambda_3$

(c)

Idx	Label
1	λ_4
2	$\neg\lambda_4$
3	λ_4

(d)

Idx	Label
1	$\neg\lambda_5$
2	$\neg\lambda_5$
3	λ_5

(e)

Table 4 Transformation by BR

Feature set	label	label	label
Fred	BMW	Volkswagen	Audi
John	Porsche	BMW	Mercedes
Andy	Mercedes	Porsche	Volkswagen
Matt	?	?	?

Table 5 An example of label ranking

The formal statement of the *label ranking* problem is to learn a mapping of instances $x \in X$ to rankings $\succ_x$ (total restrict orders) over a finite set of labels $L = \{\lambda_1, \lambda_2, ..., \lambda_c\}$, where $\lambda_i \succ_x \lambda_j$ means that for instance x, label λ_i is preferred to λ_j [27]. A ranking over L can be represented by a unique permutation τ such that $\lambda_i \succ_x \lambda_j$ iff. $\tau(\lambda_i) < \tau(\lambda_j)$, where $\tau(\lambda_i)$ denotes the position of λ_i in the tanking.

The multi-label classification problem can be related to *label ranking* as follows: each training example x is associated with a subset $P_x \subseteq L$ of possible

labels. It simply defines the set of preferences $R_x = \{\lambda_i \succ_x \lambda_j \mid \lambda_i \in P_x, \lambda_j \in L \setminus P_x\}$. The size of P_x is usually small or moderate. The labels in the set P_x are called *relevant* to the given instance; the rest are considered *irrelevant*. Approaches operate in this framework include *ranking by pair-wise comparison* (RPC)[28] and *constraint classification*[29].

The key idea of RPC is to learn, for each pair of labels (λ_i, λ_j), a binary model $M_{ij}(x)$ that predicts whether $\lambda_i \succ_x \lambda_j$ or $\lambda_j \succ_x \lambda_i$ for an input x. In order to rank the labels for a new instance, predictions for all pair-wise label preferences are obtained and a ranking that is maximally consistent with these preferences is derived. Although *constraint classification* aims at learning a linear utility function for each label, it still operates in the frame of label ranking and requires (not necessarily complete) sets of pair-wise label preferences associated with training instances to learn a ranking model which, as a post processing step, maybe projected from the label set to a specific output space.

While it is straightforward to represent the training information for multi-label classification as a preference learning problem, the algorithms which operate in the framework only produce a ranking of the available options. In order to convert the learned ranking to a multi-label prediction, the learner has to be able to autonomously determine a point at which the learned labels are split into *relevant* and *irrelevant* labels. Both RPC and *constraint classification* ignore this problem and only focus on producing rankings. The authors of [27] call this point the *zero point* and propose a conceptually new technique called *calibrated ranking*, which extends the common pair-wise learning approach to the multi-label scenario, a

setting previously not amenable to a pair-wise decomposition approach. Within this framework, RPC can solve both multi-label classification and in a consistent and generally applicable manner.

2.2.2. Simple algorithm adaption methods

This category of methods attempt to solve the multi-label classification problem by adapting algorithms which are originally applicable to single-label classification cases.

2.2.2.1. Decision tree learning

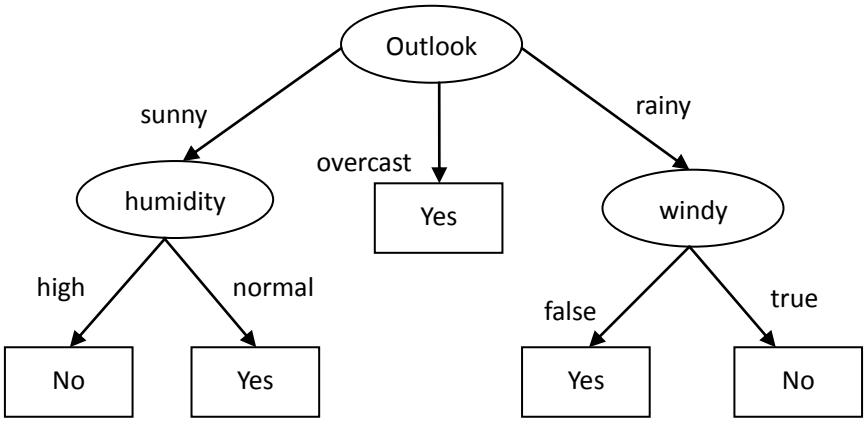
Decision tree learning is a commonly used method in data mining and machine learning, which "uses a decision tree as a predictive model which maps observations of an instance to conclusions about the instances target label"[30]. In a decision tree, internal nodes represent "conjunctions of features that lead to class labels", which are represented by leaves. Generally a decision tree works as follows: a process of splitting the instances set into subsets based on a feature attribute is repeated on each derived subset in a recursive manner until the subset at one node has all the same value as for the target label. Table 6 shows an example of decision tree derived from the data set Figure 1. (This example and corresponding figures are adapted by the author from [53].)

Due to the order of feature selection, the decision tree for one data set is not unique. Hence there are many ways to generate a decision tree. The construction and evaluation of decision trees is based on the theory of *information entropy*[31]

and *information gain*[32].

Instance	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	Hot	High	False	No
2	Sunny	Hot	High	True	No
3	Overcast	Hot	High	False	Yes
4	Rainy	Mild	High	False	Yes
5	Rainy	Cool	Normal	False	Yes
6	Rainy	Cool	Normal	True	No
7	Overcast	Cool	Normal	True	Yes
8	Sunny	Mild	High	False	No
9	Sunny	Cool	Normal	False	Yes
10	Rainy	Mild	Normal	False	Yes
11	Sunny	Mild	Normal	True	Yes
12	Overcast	Mild	High	True	Yes
13	Overcast	Hot	Normal	False	Yes
14	Rainy	Mild	High	True	No

Table 6 An example of decision tree



(b)

Figure 1 An example of decision tree

In information theory, *entropy* represents the amount of uncertainty or unpredictability contained in a random variable. It is also referred to as the *Shannon entropy*, or the information content. The average *entropy* of a discrete random variable X can be quantified as:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log_b (P(x_i))$$

Usually *b* is equal to 2, while it could also be 10 or *e* (natural logarithm).

As for how this is conceptually related to the uncertainty of information, consider the following example: let *X* be a random variable which denotes the event of raining tomorrow, and suppose the probability *P*(*X*) is 0.5. This is the maximum of uncertainty because it tells no information at all about whether or not it will rain tomorrow, since the chances of raining and not raining are equal. As can be calculated *H*(*X*) is 1 in this case. On the other hand, suppose *P*(*X*) is 0 or 1, which means it is absolutely certain that it will rain or not tomorrow. In this case the uncertainty is minimized, hence the information contained in this random variable is maximized, and *H*(*X*) is 0 (although *P*(*X*) equal to 0 or 1 is illegal in the equation, we can define the values of *H*(*X*) at these two points according to the limits). Therefore, the lower the *entropy* is, the more the information is contained in the distribution of the random variable, hence the better it can be used for future prediction.

Ross Quinlan invented an algorithm called ID3[34] to generate a decision tree based on *entropy*. It is a greedy approach which continuously takes the feature from unused features with minimum *entropy*, makes a new node and spits the instances according to that feature until all instances under one node are of the

same label. The equations of *entropy* in this case can be rewritten as follows:

$$H(S) = -\sum_{j=1}^{n} f_s(j) \log_2 f_s(j)$$

where S denotes the current instance set or subset; n denotes the number of different possible labels; $f_s(j)$ denotes the frequency of the label value j. The key idea of using *entropy* in generating a decision tree is to define a preferred sequence of feature selection which can most rapidly narrow down uncertainty.

One limitation of ID3 is that it is overly sensitive to features with large numbers of values. Consider an extreme case when there is a feature of social security number. Since everyone has a different social security number, testing on its *entropy* will always yield very low values. However, selecting social security number as a feature to split the instances obviously does not help with predicting whether a future medical patient needs a surgery.

To overcome this problem Quinlan invented another algorithm called C4.5[35], which is an extension of ID3. The framework of C4.5 to generate a decision tree is exactly the same as ID3, however it introduces a metric as for how a feature is selected, which is based on the concept of *information gain*, which is defined by subtracting the *conditional entropy* from the *base entropy*:

$$IG(S,a) = H(S) - \sum_{i=1}^{m} f_s(a_i) H(S_{a_i})$$

where $H(S)$ is the *base entropy* and $\sum_{i=1}^{m} f_s(a_i) H(S_{a_i})$ is the *conditional entropy*; S denotes the current instances set or subset; for one chosen feature a, $IG(S,a)$ denotes the *information gain* produced by a split over the feature a; m

is the number of different values of feature a; $f_s(a_i)$ is the frequency of the items possessing a_i as value for a in S; a_i is the ith possible value for a; S_{a_i} is a subset of S containing all items where the value of a is a_i. This computation does not, in itself, produce anything new. However it allows to measure the *gain ratio*, defined as $GainRatio(S \mid S_a) = IG(S,a) / H(S_a)$, where $H(S_a)$ is the *entropy* of instances only relative to feature a. It measures the *information gain* of feature a relative to the "raw" information of the S_a distribution. By using *gain ratio* instead of plain *conditional entropy*, C4.5 reduces the problem of artificially low *entropy* values such as was seen with social security number.

The decision trees generated by C4.5 are statistical classifiers for single-label classification, as each instance will be ultimately labeled as belonging to one class. Clare et. al.[36] extends the C4.5 algorithm in order for multi-label classification by modifying the *entropy* calculation:

$$H(S) = -\sum_{j=1}^{n} \{ f_s(j) \log_2 f_s(j) + (1 - f_s(j)) \log_2 (1 - f_s(j)) \}$$

where $f_s(j)$ denotes the frequency of the label value j.

Entropy is a measure of the amount of uncertainty in the dataset. It can also be thought in another way: given an instance of the dataset, how much information is needed to describe that instance? This is equivalent to asking how many binary bits are needed to describe all the labels it belongs to. The alternated formula shown above is a sum of the number of bits needed to describe

membership or non-membership of each label.

To illustrate this idea, consider a bit string with four labels: $\{a,b,c,d\}$. An instance belonging to label b and d could be represented by four bits 0101. However, this is more than enough if we know the distribution of the labels. For example, what if we already know that every instance belongs to label b? Then the second bit could simply be dropped and only three bits are needed. In other words, we need 0 (log 1) bits to represent if an instance belongs to label b. What if we know 75% of the instances belong to label b? Then we know intuitively an instance is more likely to belong to label b than not. The amount of information gained by actually knowing whether a particular instance belongs to label b or not will be $(\log 1-\log 0.75)$ and $(\log 1-\log 0.25)$, hence the expected amount of information gained is:

$$0.75\times(\log 1-\log 0.75)+0.25\times(\log 1-\log 0.25)=0.81$$

It means we actually only need 0.81 bits to represent the information about the membership or non-membership of label b for an instance. This rule can be generalized for the whole label set, which leads to the *entropy* formula for multi-label classification introduced earlier. With the alternated *entropy*, it also has to allow leaves of the trees to potentially be a set of labels, i.e. the outcome of a classification of an instance can be a set of labels.

2.2.2.2. Boosting

Boosting is another powerful technique for machine learning. The basic idea of *boosting* is to combine a series of 'base' classifiers to produce a 'committee'

whose overall performance can significantly outperform any of the base classifiers, even the base classifiers (also called *weak learners*) only perform slightly better than random. *Boosting* was originally meant to solve classification problems, it can also be adapted to solve regression problems.

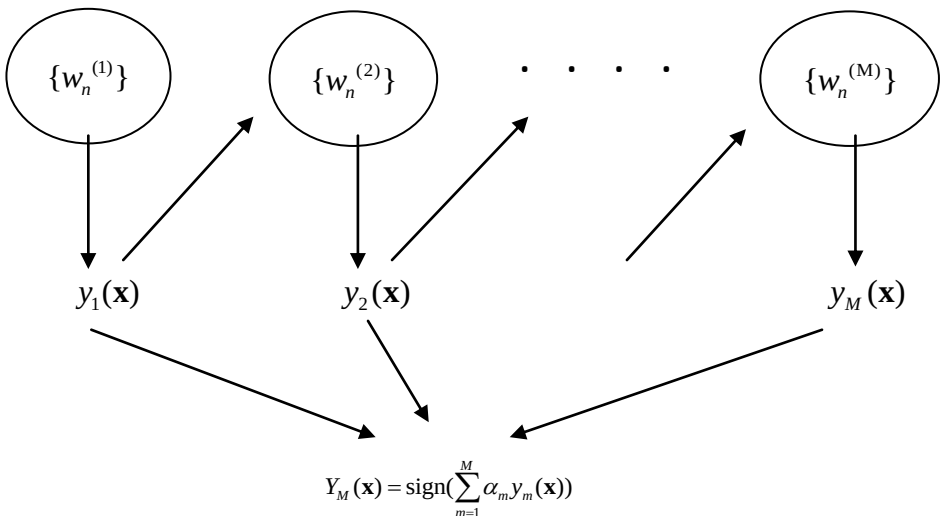


Figure 2 Framework of AdaBoost

There are other machine learning methods which also construct a committee to take the average predictions of a group of individual models, such as *bootstrap bagging*[37]. The major improvement of *boosting* over such methods is that in *boosting*, the individual base classifiers are trained in sequence. Moreover, each base classifier is associated with a data point set in which each data corresponds to a weighting coefficient that is iteratively adjusted according to the performance of the previous classifiers. The principal updating idea is that if a point is classified wrongly by the current base classifier, its weight will increase when the next classifier in the sequence is trained. The learning phase is finished after all the base classifiers are trained. In the generalization phase, the label of a new

instance is decided by taking the weighted majority of votes from the base classifiers. The basic framework of boosting is shown in Figure 2, which is adapted by the author from [45].

The most widely used *boosting* algorithm is *AdaBoost* (*Adaptive Boosting*)[38]. Consider a binary label classification problem: training data instance consists of input feature vectors $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ with corresponding binary labels $t_1, t_2, ..., t_N$ where $t_n \in \{-1, 1\}$. Each instance, which can be seen as a data point, is associated with a corresponding weighting coefficient w_n. Suppose a base classifier is already available, which maps an instance to one of the two labels: $y(\mathbf{x}) \in \{-1, 1\}$. The precise form of *AdaBoost* algorithm is given below:

1. Initialize the data weighting coefficients $\{w_n\}$ by setting $w_n^{(1)} = 1/N$ for $n = 1, 2, ..., N$.
2. For $m = 1, 2, ..., M$:
   a) Train a classifier $y_m(\mathbf{x})$ with the training data by minimizing the weighted error function:

$$E_m = \sum_{n=1}^{N} w_n^{(m)} F(y_m(\mathbf{x}_n), t_n)$$

   where $F(y_m(\mathbf{x}_n), t_n) = \begin{cases} 0, & y_m(\mathbf{x}_n) = t_n \\ 1, & y_m(\mathbf{x}_n) \neq t_n \end{cases}$

   b) Evaluate $\varepsilon_m = \dfrac{E_m}{\sum_{n=1}^{N} w_n^{(m)}}$ and then compute $\alpha_m = \ln\{\dfrac{1-\varepsilon_m}{\varepsilon_m}\}$

   c) Update the weight coefficients as follows:

$$w_n^{(m+1)} = w_n^{(m)} \exp\{\alpha_m F(y_m(\mathbf{x}_n), t_n)\}$$

3. Make predictions on new instances as follows:

$$Y_M(\mathbf{x}) = \text{sign}(\sum_{m=1}^{M} \alpha_m y_m(\mathbf{x}))$$

As can be seen, a new base classifier is trained with a data set whose weighting coefficients are updated based on the performance of the previously trained base classifier so that former misclassified data points are given greater weight.

AdaBoost has been studied extensively and has been shown to perform well on standard machine-learning tasks. Schapire[39] describes how this algorithm can be extended and generalized in order to address text-categorization task, which is usually a multi-class, multi-label classification problem. Two extensions of the *AdaBoost* algorithm are specifically intended for multi-class, multi-label data: the first extension tries to predict a good approximation of the set of labels associated with a text document; and then the second extension tries to rank the labels so that the correct labels will receive the highest rank.

The first extension is called *AdaBoost.MH*. It is actually a natural reduction of the multi-class, multi-label data to binary data, which has been introduced earlier (*Binary Relevance*): each training instance (x, Y) is mapped to *k* binary labeled examples, where *k* is the size of all possible labels, depending on whether or not a certain label is in the current label set *Y*. Then the original binary *AdaBoost* can be applied to train the derived binary data. The space and time per-round (each call of *weak learner*) complexity is $O(mk)$, where *m* is the size of training set. Similar

to the original *AdaBoost* algorithm, *AdaBoost.MH* maintains a weight distribution over $X \times Y$, and adjusts the weights at each boosting step so that instances that are misclassified by the hypothesis in the previous round have a higher weight in the current round.

It is still unclear how to quantify the error. In single label classification this is simple as there is only one output label for one instance, therefore either it is completely right or completely wrong. It is more complicated in multi-label classification as the classification can be "partly" correct. There are a couple of ways to evaluate the error depending on the specific application to deal with. Here *AdaBoost.MH* considers *Hamming Loss*, which takes into account *prediction errors* (an incorrect label is predicted) and *missing errors* (a correct label is not predicted). Suppose the hypothesis function is $h: X \mapsto 2^Y$, the *Hamming Loss* error over a training sample set *S* is defined by:

$$E_H(h,S) = \frac{1}{km}\sum_{i,l}(\| l \in h(x_i) \cap l \notin Y_i \| + \| l \notin h(x_i) \cap l \in Y_i \|)$$

where the factor $\frac{1}{k}$ normalizes the error in the interval [0, 1]; and $\|a\|$ equals 1 if a holds and 0 otherwise. This idea can be further generalized to evaluate prediction error for new unforeseen instances, if a target function $c: X \mapsto 2^Y$ is known.

The second extension is called *AdaBoost.MR*. It bears the same framework as the first extension with a different goal to minimize the average fraction of *misordered crucial pairs* which are relative orderings of l_0, l_1, for which one of them is in the current label set while the other is not. Suppose with respect to a

labeled observation (x, Y), $l_0 \in Y$ and $l_1 \notin Y$, a classification rule f *misorders* the crucial pair if $f(x, l_1) \leq f(x, l_0)$ so that f fails to rank l_1 over l_0. The space and time-per-round complexity is the same as the first extension.

2.2.2.3. *k* Nearest Neighbor

Both *decision tree* based and *boosting* based approaches try to construct a general hypothesis function solely based on learning the training set, without any knowledge about the input during the generalization phase. Such methods are called *eager learning* in artificial intelligence. In contrast, there are methods called *lazy learning* in which "generalization beyond the training data is delayed until a new query of unforeseen instance is made for classification"[41]. The advantage of *lazy learning* is that the hypothesis function is approximated locally, therefore the new query is more closely correlated to some particular training data instances. The disadvantage is that since the function abstraction is limited on a local scale concerning only a small group of data points, noise or abnormal instances can sometimes significantly affect generalization performance. And for the same reason it requires large space to store enough training data in order to achieve good performance. This also leads to the fact that *lazy learning* methods have a shorter training phase, but take longer to generalize. Therefore *lazy learning* are most useful for large data with few features.

The representative of *lazy learning* approach is *k Nearest Neighbor* (*k*-NN). It classifies new instances based on closest training examples in the feature space: the label of a new instance is determined by the votes of its neighbors, that is, the

instance is assigned the most common label among its *k* nearest neighbors, where *k* is an small integer. The *k*-NN algorithm is one of the simplest machine learning algorithms[42].

There are *k*-NN based approaches that have been proposed for multi-label classification in combination with either problem transformation or algorithm adaption introduced earlier. For example, BR*k*NN conceptually uses *Binary Relevance* to transfer the problem and then takes *k*-NN method as a classifier. The author in [43] points out two possible problems of directly combining the implementation of them. One of them is that simply applying *k*-NN on the basis of *Binary Relevance* would incur a time cost |*L*| times that of *k*-NN algorithm, where *L* is the size of all possible labels. This could be crucial in domains with a large set of labels and strict requirements for response time. Another problem is that since *Binary Relevance* trains every label independently, it is possible that an instance turns out not to belong to any label. The author then proposes two extensions of BR*k*NN to address these two problems.

2.2.2.4. Support Vector Machine

Support Vector Machine (SVM)[20] is another supervised learning approach for both classification and regression. An SVM training algorithm builds a model which is an representation of the examples as points in space, marked as belonging to one of two categories. The example points are then mapped in such a way that separate categories are divided by a clear gap that is as wide as possible. In the generalization phase, new examples are predicted to belong to a category based on which side of gap they fall on.

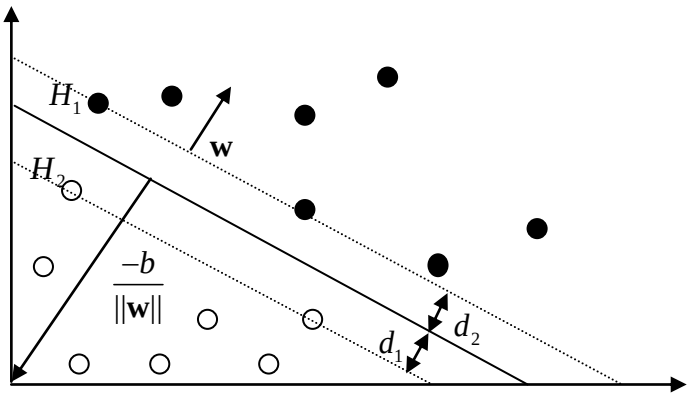


Figure 3 Support Vector Machine

Suppose S is the training set, whose size is m, in which each input $\mathbf{x}_i$ has n attributes, therefore can be seen as a point in n dimensional space and is labeled as two classes: $y_i = -1$ or $y_i = -1$. Assume the data is linearly separable, which means a line can be drawn on the graph of x_1 vs. x_2 separating two classes when $n = 2$ and a hyperplane on graphs of $x_1, x_2, ..., x_A$ for when $n > 2$, shown in Figure 3 (this example and the corresponding figure are adapted by the author from [54]).

The hyperplance can be described by $\mathbf{w} \cdot \mathbf{x} + b = 0$ where w is normal to the hyperplane and $\frac{-b}{||\mathbf{w}||}$ is perpendicular distance from the hyperplane and the origin. *Support Vectors* are example points closest to the separating hyperplane and the aim of *Support Vector Machine* is to orientate this hyperplane so that it is as far as possible from closest members from both classes. As shown in Fig. 8, implementing a SVM can be reduced to the selection of $\mathbf{w}$ and b to fix the

hyperplane so that:

$$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1 \quad \text{for } y_i = +1$$
$$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1 \quad \text{for } y_i = -1$$

which can be combined into one formula:

$$y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \quad \forall i$$

Support Vectors are the points which lie on two planes H_1 and H_2. The distances from H_1 and H_2 to the hyperplane are d_1 and d_2, which are equivalent to each other. This distance is called SVM's *margin*. To orientate the hyperplane so that the *Support Vectors* are as far away as possible means SVM's *margin* needs to be maximized.

It can be shown by vector geometry that the SVM's *margin* is equal to $\frac{1}{\|\mathbf{w}\|}$.

Maximizing this objective function with constraints that the hyperplane separates points of distinct labels leads to:

$$\min \|\mathbf{w}\| \quad \text{such that} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \forall i$$

This can be transformed into a *Quadratic Programming* problem:

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{s. t.} \quad y_i(\mathbf{x}_i \cdot \mathbf{w} + b) - 1 \geq 0 \ \forall i$$

To solve this we need to minimize:

$$L_P \equiv \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i y_i (\mathbf{x}_i \cdot \mathbf{w} + b) + \sum_{i=1}^{m} \alpha_i$$

where α_i are Lagrange multipliers and $\alpha_i \geq 0$. Taking derivatives on $\mathbf{w}$ and b, and then substituting the results ($\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ and $\sum_{i=1}^{m} \alpha_i y_i = 0$) back we have:

$$\max L_D \equiv \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j \quad \text{s. t. } \alpha_i \geq 0 \quad \forall i, \sum_{i=1}^{m} \alpha_i y_i = 0$$

which leads to:

$$\max_{\alpha} [\sum_{i=1}^{m} \alpha_i - \frac{1}{2} \boldsymbol{\alpha}^T \mathbf{H} \boldsymbol{\alpha}] \quad \text{s. t.} \quad \alpha_i \geq 0 \quad \forall i, \sum_{i=1}^{m} \alpha_i y_i = 0$$

where $H_{ij} \equiv y_i y_j \mathbf{x}_i \mathbf{x}_j$. This is a convex quadratic optimization problem, therefore a QP solver can be run to return α, and then $\mathbf{w}$ can be deducted from the former derivative conditions. To determine b, a *Support Vector* has the form $y_s(\mathbf{x}_s \cdot \mathbf{w} + b) = 1$, and therefore:

$$y_s(\sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s + b) = 1$$

where S denotes the set of indices of the *Support Vectors*. S is determined by finding the indices i where $\alpha_i > 0$. Since $y_s^2 = 1$, we have:

$$y_s^2(\sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s + b) = y_s$$
$$b = y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s$$

Instead of using an arbitrary *Support Vector* $\mathbf{x}_s$, it is better to take an average over all of the *Support Vectors* in S:

$$b = \frac{1}{N_S} \sum_{s \in S} (y_s - \sum_{m \in S} \alpha_m y_m \mathbf{x}_m \cdot \mathbf{x}_s)$$

Once $\mathbf{w}$ and b are determined, the hyperplane is fixed. For a new instance point $\mathbf{x}'$ is classified by evaluating $y' = \text{sgn}(\mathbf{w} \cdot \mathbf{x}' + b)$. This algorithm can be extended to handle the data that is not fully linearly separable with simple adaption.

The matrix $\mathbf{H}$ which plays a crucial role in the algorithm is created from the

dot product of the input variables:

$$H_{ij} = y_i y_j k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j = \mathbf{x}_i^T \mathbf{x}_j$$

$k(\mathbf{x}_i, \mathbf{x}_j)$ is an example of a family of functions called *Kernel Functions* ($k(\mathbf{x}_i, \mathbf{x}_j)$ is known as the *Linear Kernel*). The set of kernel functions is composed of such functions which are all based on calculating inner products of two vectors. The idea is that if the functions can be recast into high dimension space by some non-linear feature mapping function $\mathbf{x} \mapsto \phi(\mathbf{x})$, only inner products of the mapped inputs in the feature space need to be determined without explicitly calculating ϕ. This is called the *Kernel Trick*[54].

This *Kernel Trick* is useful to deal with classification problems that are completely not linearly separable in the space of the inputs $\mathbf{x}$, as they might be separable in a higher dimensionality feature space given a suitable mapping function $\mathbf{x} \mapsto \phi(\mathbf{x})$. There are a few kinds of kernel functions. For example, $k(\mathbf{x}_i, \mathbf{x}_j) = e^{-(\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})}$ is known as *Radial Basis Kernel*; $k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + a)^b$ is known as *Polynomial Kernel*; $k(\mathbf{x}_i, \mathbf{x}_j) = \tanh(a\mathbf{x}_i \cdot \mathbf{x}_j - b)$ is known as *Sigmoidal Kernel*, where a and b are parameters that define the kernel's behavior. By means of kernel functions the non-linearly separable input data can be made separable, and therefore the SVM method can be applied.

The SVM method can also be combined with BR serving as a basic binary classifier like KNN in a similar way as introduced before. Several ideas are proposed in [46] to improve the *margin*. First the dataset is extended to have

additional features which are actually predictions of each binary classifier at the first round. The new binary classifiers are trained on the extended dataset so that the extended BR takes into account potential label dependencies. And then negative training examples of a complete label are removed if it is very similar to the positive label. Finally similar negative examples within a threshold distance from the learned decision hyperplane are also removed to build better models especially in the presence of overlapping classes.

Chapter 3

Multi-label Bayesian Decision Trees

As introduced in Chapter-1, the objective of BrainMap Tracker is to annotate the cognitive neuroimaging papers according to CogPO. The essence of this task is still text mining, or more specifically, document classification, but it is also different from normal cases that the target annotations are related to each other and organized in a pattern predefined by the ontology.

We analyzed the results of the NCBO annotator, which also serves as our baseline of development shown in the Chapter-4, and found that its performance on different CogPO components differ significantly. The NCBO annotator captures ontological terms for certain components (for example, Stimulus Type) quite effectively, while for some components it achieves almost nothing (for example, Stimulus Modality). This indicates that the internal difficulty to capture the terms corresponding to different components also differs significantly.

An important reason is that the current NCBO annotator is based on pure text matching. It often fails to identify the correct ontological term when there are a number of possible alternatives available for describing a concept, and also when a concept is hard to be explicitly expressed in one single word and therefore the intent of the author is usually hidden deep in the meaning of the context . This variety among different components indicates a decision tree might be useful because we can prioritize those components which tend to give us accurate predictions and effectively narrow down the scale of the problem. On the other hand, in regards to the existing classifying approaches, *Naive Bayes* is

considered the best for document classification[47]. On the basis of these intuitions we propose a combined model of decision tree and *Naive Bayes* classifier for solution of the BrainMap Tracker.

This chapter is the result of a group effort. The basic framework of the algorithm is mainly proposed and implemented by my colleague, Thomas B. Jones. The author and another colleague, Chayan Chakrabarti, contributed to the solution by taking part in the group discussions and the algorithm implementation. In particular, the author implemented a component which evaluates the performance of our approach in terms of f-scores, and helped my colleagues by correcting a few bugs during programming.

3.1. Naive Bayes classifier

Naive Bayes classifier is the simplest instance of a probabilistic classifier. It is based on the assumption that for a given class the features of the class are independent of each other. The model can be represented as $p(C \mid F_1,....,F_n)$, where C is the class variable, conditional on features F_1 through F_n. According to the Bayes' theorem:

$$p(C \mid F_1,....,F_n) = \frac{p(C)\,p(F_1,....,F_n \mid C)}{p(F_1,....,F_n)}$$

The denominator is effectively constant since it does not depend on C and the values of F_1 through F_n are given. Therefore the numerator is effectively equivalent to the joint probability model $p(C, F_1,....,F_n)$, which can be repeatedly rewritten as follows:

$$
\begin{aligned}
p(C, F_1, ..., F_n) &\propto p(C)\, p(F_1, ..., F_n \mid C) \\
&\propto p(C)\, p(F_1 \mid C)\, p(F_2, ..., F_n \mid C, F_1) \\
&\propto p(C)\, p(F_1 \mid C)\, p(F_2 \mid C, F_1)\, p(F_3, ..., F_n \mid C, F_1, F_2) \\
&..... \\
&\propto p(C)\, p(F_1 \mid C)\, p(F_2 \mid C, F_1) .... p(F_n \mid C, F_1, F_2, ..., F_{n-1})
\end{aligned}
$$

Now the 'naive' feature independence assumption comes into play: if F_i is conditionally independent of F_j, as long as $i \neq j$ given the class C, it means:

$$p(F_i \mid C, F_j) = p(F_i \mid C)$$

for $i \neq j$, therefore the formal expression of the joint model can be further written as:

$$
\begin{aligned}
p(C \mid F_1, ..., F_n) &\propto p(C)\, p(F_1 \mid C)\, p(F_2 \mid C) .... p(F_n \mid C) \\
&\propto p(C) \prod_{i=1}^{n} p(F_i \mid C) \\
&= \frac{1}{Z}\, p(C) \prod_{i=1}^{n} p(F_i \mid C)
\end{aligned}
$$

where Z is a scaling factor dependant only on $F_1, ..., F_n$, hence a constant if the value of the feature variables are known.

The *Naive Bayes* classifier applies a decision rule to this probability model. One common rule is to select the hypothesis that has the highest probability, which is known as *maximum a posteriori* (MAP) decision rule. The classifier works as follows:

$$\text{classify}(f_1, ..., f_n) = \arg\max_c\, p(C = c) \prod_{i=1}^{n} p(F_i = f_i \mid C = c)$$

In spite of the oversimplified assumption, naive Bayes classifiers work surprisingly well in many real world situations because of several properties. In particular, the decoupling of the class conditional feature distributions means each distribution

can be independently estimated as a one dimensional distribution. This helps alleviate problems stemming from dimensionality, such as the need for data sets that scale exponentially with the number of features.

3.2. Naive Bayes for document classification

All the machine learning methods introduced above have been applied to address the challenge of automatic document classification. Among them, *Naive Bayes* text classifier has been widely used because of its simplicity in both the training and the classifying stages. It allows each feature attribute to contribute toward the final decision equally and independently from other feature attributes, which makes it computationally more efficient compared to other text classifiers.

A typical framework to generate a document classifier model is shown below in Figure 4, which is adapted by the author from [47]. The input dataset is the raw documents, each of which consists of a set of words serving as feature attributes. All words ought to be found from a 'dictionary', which can be considered as the whole feature space.

The whole process begins with data preprocessing with the model evaluation, which usually involves removing stop words and stemming. Removing stop words means taking out of the words from the document whose presence is necessary for grammatical correctness but contains no substantial information and hence useless for classifying the document. Such words could be 'a', 'the', 'in', 'at', and so forth. Stemming means combining words which carry similar meanings but in different grammatical forms into one attribute. For example, 'soldiers' is the plural

form of 'soldier', but both of them describe the same entity.

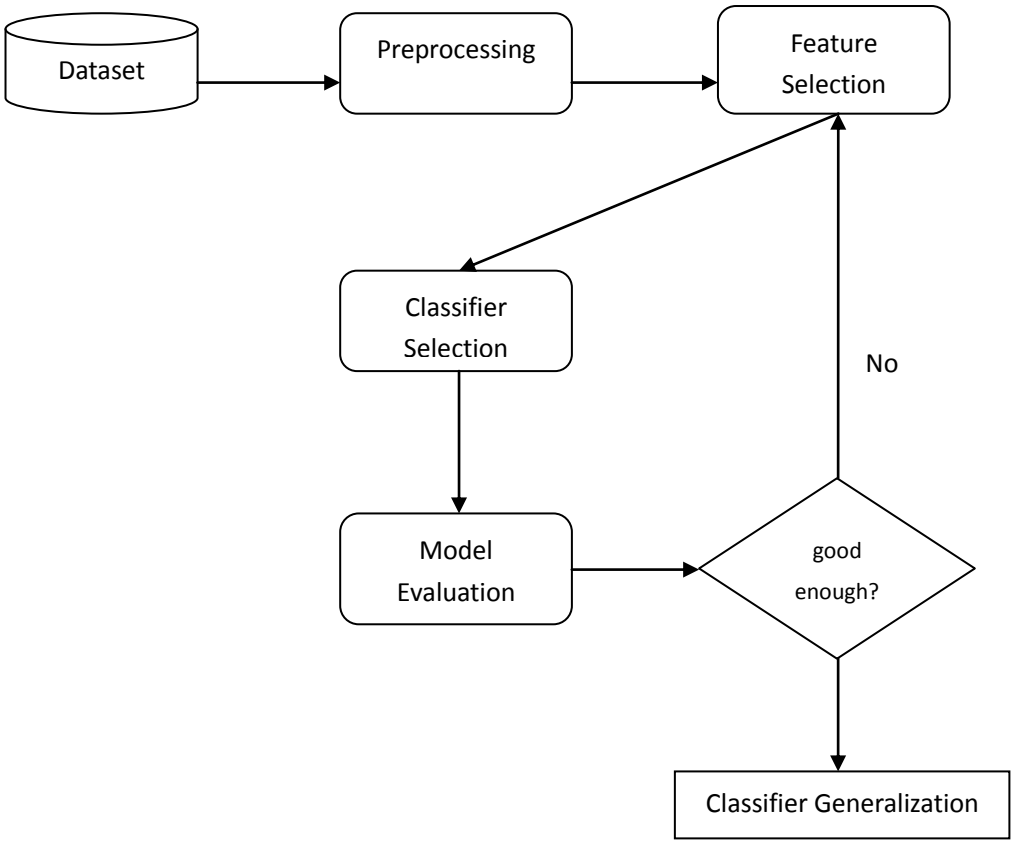


Figure 4 Flow of document classification

The next step after preprocessing is feature selection, which is one of the most important steps for data mining. It means selecting a subset of the feature attributes which are relevant to a given analysis task. The general strategy is to select subsets, learn a model on the subset and evaluate the performance of the learned model. The subset on which the highest performance is achieved is then selected as input to the subsequent steps. There are different ways as for how subsets are selected. For example, the *brute force* strategy simply evaluates the performance of all possible subsets; and *forward selection* uses one attribute at

the beginning and adds additional attributes heuristically until performance is no longer improved. Several feature selection methods specifically dedicated to text mining have been proposed and studied[48][49].

After preprocessing and feature selection, the numbers of feature attributes should reduce significantly. The next step is to apply the classifier to the dataset. Since it is a general workflow of document classification, we can actually apply any classifier here, like *Nearest Neighbor*, *Decision Tree*, or SVM. Here we focus on *Naive Bayes*, which is used because of its simplicity and good performance in text and document classification[50].

The way naive Bayes classifier works with document classification is as follows: consider the problem of classifying a document into a class C or $\neg C$. A document can be modeled as a set of words where the probability that the i th word of a given document occurs in a document classified as C can be written as $p(w_i \mid C)$. Here we assume that words are randomly distributed in the document, which means they are independent of the length of the document, and position within the document with respect to other words. Therefore the probability that a given document D contains all the words w_i, given a class C is $p(D \mid C) = \prod_i p(w_i \mid C)$, while what we are interested in is the probability of a given document belonging to a class, which is $p(C \mid D)$. According to Bayes' theorem:

$$p(C \mid D) = \frac{p(C)}{p(D)} p(D \mid C)$$

Since this is a binary classification problem, there are only two classes C and $\neg C$, therefore:

$$p(C \mid D) = \frac{p(C)}{p(D)} \prod_i p(w_i \mid C)$$

$$p(\neg C \mid D) = \frac{p(\neg C)}{p(D)} \prod_i p(w_i \mid \neg C)$$

Combining them we have:

$$\frac{p(C \mid D)}{p(\neg C \mid D)} = \frac{p(C)}{p(\neg C)} \prod_i \frac{p(w_i \mid C)}{p(w_i \mid \neg C)}$$

The exact value of $p(C \mid D)$ and $p(\neg C \mid D)$ can be computed because of the fact that $p(C \mid D) + p(\neg C \mid D) = 1$. For classification, if $p(C \mid D) > p(\neg C \mid D)$ then document D is classified as C, otherwise it does not belong to C. This basic naive Bayes classifier for binary classification can be extended to multi-class, multi-label cases using the problem transformation methods introduced before.

After classification the technique of *F-Measure* introduced earlier can be applied to evaluating the performance of the model. If it is not good enough, the model can be adjusted by repeating the process from feature selection to model evaluation again, until a satisfactory result is obtained.

3.3. Multi-label Bayesian Decision Tree

On the basis of the naive Bayes classifier and the decision tree, we propose a combined model, called *Multi-label Bayesian Decision Tree*. It first trains a naive Bayes classifier on each component of CogPO, and then depending on the performance of the classifiers a decision tree is built, which decides the label for each component in the order of prediction confidence. The idea is to take advantage of the components whose labels are easy to classify, narrow down the

input dimension space, in order to help with those components whose labels are difficult to decide. This strategy exploits the internal relationship among the component labels to improve the accuracy of classification

Suppose we have a training set S, composed of abstracts annotated for each component in the component set C by human experts. We train a multi-label naive Bayes classifier $B_{c,S}$ on each component c. That is, $B_{c,S}$ only focuses on what labels the abstracts are annotated for component c, and ignores the rest components. In order to test the performance for each of the classifiers $B_{c,S}$ we split the training examples into k subsets randomly and then use one of them for evaluation purpose and the rest for training the classifier. This process is repeated for k times until each subset has been used as a testing set, and then the performance of the classifier is taken from the average of the k trials. This is called *K-fold cross validation*[51]. We apply *K-fold cross validation* to every component, therefore obtain C refined naive Bayes classifiers corresponding to each component.

Suppose the refined naive Bayes classifiers are ordered by performance in terms of f-score from highest to lowest: $B_{c_1,S}, B_{c_2,S}, \ldots, B_{c_m,S}$, where m is the size of the component set C. The construction of a multi-label decision tree works as follows: each tree node represents the component labels that have already been decided. Therefore the root has no labels because nothing has been decided yet. To start we pick the classifier which has the highest f-score and associate it with the root of the tree, $B_{c_1,S}$. The root has n_{c_1} children, where n_{c_1} corresponds to the size of the label set of the component c_1, and each child

corresponds to one label of the label set. Then each child is associated with a classifier $B_{c_2,S}$, trained to classify component c_2, with a subset of the training examples, in which each abstract has component c_1 classified as the label that the current child contains (It is possible that an abstract has multiple labels for one component. In this case as long as the label represented by the current child is contained, the abstract is included in the training set). In this way the labels for component c_2 are decided. Each child then has n_{c_2} children, and n_{c_2} is the size of the label set of the component c_2. Each of these n_{c_2} children has two labels, one for component c_1 and the other for component c_2, and associated with another basic, multi-label naive Bayes classifier, trained by the example abstracts containing labels of the current child for component c_1 and c_2, in order to classify component c_3. Therefore on this level labels for c_3 are decided. This process is repeated until labels for each component are decided and finally a decision tree of $|C|$ levels is constructed. Here the training phase is completed.

In the generalization phase, when classifying a new abstract, we first use the multi-label naive Bayes classifier associated with the root, $B_{c_1,S}$, to decide the labels of component c_1. Suppose it returns a label set C_1 which contains labels $l_1, l_2, ..., l_d$. We traverse down the tree to the d children corresponding to the labels. On the next level, we obtain predictions for classifying component c_2, based on the conditions of the labels decided for c_1, and then go down one more level to classify c_3 with the information of c_1 and c_2, and so forth. On the bottom

level we collect all the labels on the traversed paths and the classification is complete. The formal description of the algorithm is as follows:

Algorithm Multi-Label Bayesian Decision Tree:

Input: an unclassified document D, a Multi-Label Bayesian Decision Tree T

Output: label vector in multiple components L_D

$t = Root(T)$

$SearchList = NULL$

while $t \neq NULL$

$L_D = L_D : B_t(D)$

for $l \in B_t(D)$

$SearchList = SearchList : Child(l, t)$

end

$t = SearchList[0]$

$x : SearchList = SearchList$

end

return L_D

In our case, the components set has CogPO's five components: stimulus type, stimulus modality, response type, response modality and instructions:

$$C = \{ST, SM, RT, RM, I\}$$

Each of them has a different set of possible labels. The complete lists of these five label sets can be found in **Appendix B**.

Chapter 4

Experimental Results

To build the training data set we select 247 abstracts of academic research papers in the brain mapping field which have been annotated manually with CogPO (Cognitive Paradigm Ontology). This collection of abstracts is organized as a table called the corpus, in which each entry corresponds to one abstract, containing its basic information such as the identification number (ID) in the PubMed database, what experiments it deals with, when and in which journal it is published, who the author is, and the annotation results which consist of ten CopPO components: "Diagnosis", "Stimulus Modality", "Stimulus Type", "Response Modality", "Response Type", "Instruction", "Context", "Paradigm Class", "Behavioral Domain", and "Prose Description". Each component contains one or more terms as its labels. The most crucial five of them (Stimulus Modality, Stimulus Type, Response Modality, Response Type, Instruction) are selected for our automatic classification task. A full list of the abstracts with information of these five components can be found in **Appendix A**.

This chapter is the result of a group effort. Section 4.1 is put together from the results of NCBO annotator and the author's implementation of f-score evaluation. The statistics in section 4.2 for MLBT result from the implementation of my colleague Thomas B. Jones. The adaption of MLBT* is implemented and tested by the author.

4.1. Baseline

The collection of abstracts including their titles and keywords are annotated by NCBO's standard annotation tool using CogPO. The results of CogPO are collected and compared with the manual annotations, as shown in Figure 5.

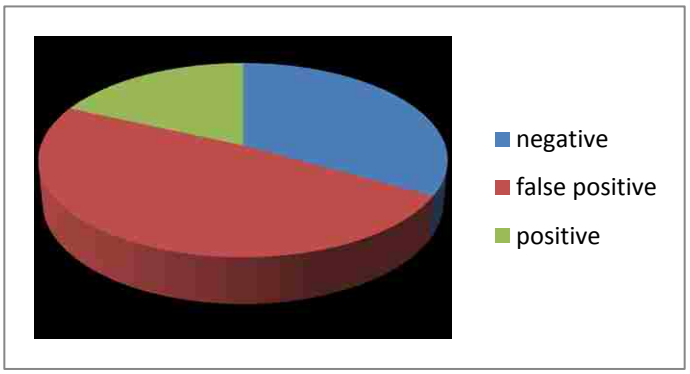


Figure 5 Annotation results from NCBO

Among those 165 annotation results, 120 of them fail to produce any term that matches the manual annotation. This means the terms returned by the NCBO annotation are not found in the corresponding manual annotations. These 120 results are considered as false positive. For instance, the annotation result of the abstract with ID 9862553 contains a term "function". This term is searched in the corresponding entry in the corpus among the components of CogPO ontology listed above. If none of the field values contains the word function, it is considered as a false positive result. Some of such terms are common words like "function", which have little to do with the ontology; while some of them are very likely ontology terms but without a match in the corpus, like "speech", which could be the value of "Response Type".

Therefore, 45 of the 165 non-empty annotation results produce at least one

term that matches the corresponding manual annotation. This means the term could be found in the content of one or more of the manually annotated CogPO components, which are “Diagnosis”, “Stimulus Modality”, “Stimulus Type”, “Response Modality”, “Response Type”, “Instruction”, “Context”, “Paradigm Class”, and “Behavioral Domain”. Since one abstract can have multiple different annotation terms, it can also have multiple of them correctly annotated. For instance, abstract of ID 16497485 has term "Recall" and "Words" both correctly annotated. Actually this is the only case. Therefore there are 46 correct annotations in total. On the other hand one correctly annotated term can match multiple CogPO components simultaneously. For instance, abstract of ID 10080553 has the term "recall" correctly annotated which could be found in both "Instruction" and "Paradigm class".

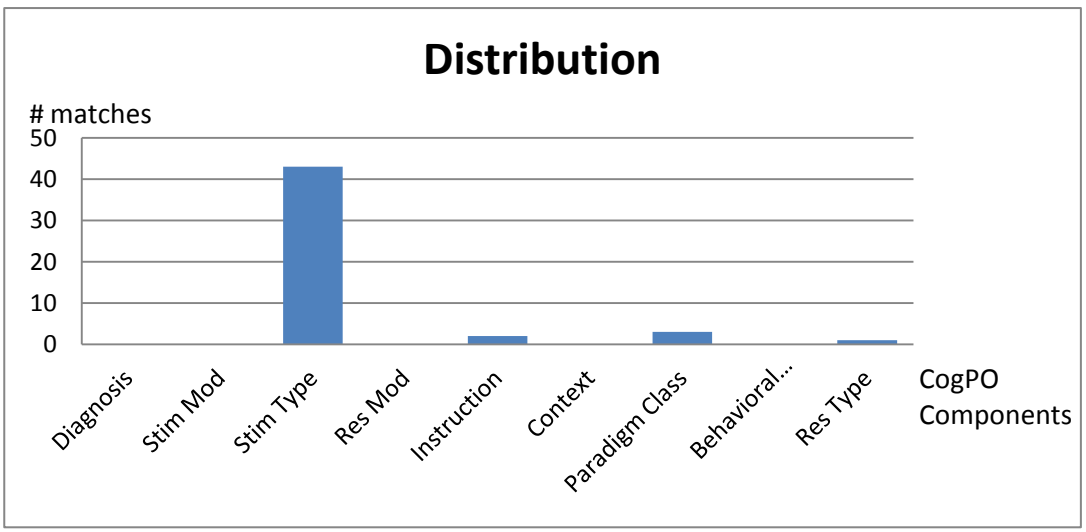


Figure 6 Distribution of correct annotations

The distribution of the terms among these components is in Figure 6. As can be seen, most terms correctly annotated come back from “Stimulus Type”. Such terms are usually “words”, “pictures” or “faces”. In comparison with false positive

terms, the set of correctly annotated terms is much smaller, which has only 6 terms.

Among all CogPO components the most important ones which decide the accuracy of an annotation attempt are "Stimulus Modality", "Stimulus Type", "Response Modality", "Response Type" and "Instruction". Each annotation result covers predictions for all of these five components, and each of them has a distinct set of terms as for what an annotation result could be. The calculated F-measure scores for each of those components are listed below:

	Instruction	Stimulus Modality	Stimulus Type	Response Type	Response Modality
F-measure	0.00425	0	0.00941	0.00223	0

Table 7 F-Scores of NCBO annotation

ST>SM>RT>RM>I	Instruction	Stimulus Modality	Stimulus Type	Response Type	Response Modality
F-measure	0.495	0.816	0.463	0.689	0.757
Standard deviation	0.126	0.067	0.133	0.114	0.068

Table 8 F-Scores of Multi-label Bayesian Decision Tree

4.2. Performance of The Multi-label Bayesian Decision Tree

We implemented our Multi-label Bayesian Decision Tree (MLBT) on the basis of the multi-label naive Bayes classifier provided by Mallet (Machine Learning for Language Toolkit)[52], a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text. For *K-fold cross*

validation we set $k = 5$. The components are ordered as {ST, SM, RT, RM, I} so that *Stimulus Type* is set as the root of decision tree. The performance on the same benchmark is shown in Table 8.

Comparing Table 8 with Table 7, it is obvious MLBT significantly outperforms NCBO's annotator. For components Stimulus Modality and Response Modality, it produces correct labels in most cases; for the rest three components which have large label sets, it makes the right decisions in a fair amount of cases. On the other hand, NCBO's annotator achieves almost nothing. **Appendix B** lists F-scores of each label in each component.

There is much room to improve on the basis of NCBO's annotator, and it does not necessarily justify our approach. To show MLBT is an effective solution, we need to look further into the alternatives.

	Instruction	Stimulus Modality	Stimulus Type	Response Type	Response Modality
SM>ST>RT>RM>I	0.476	0.828	0.461	0.678	0.735
RT>ST>SM>RM>I	0.487	0.791	0.426	0.726	0.769
RM>ST>SM>RT>I	0.484	0.778	0.420	0.714	0.776
I>ST>SM>RT>RM	0.519	0.830	0.406	0.686	0.759
ST>SM>RT>RM>I	**0.495**	**0.816**	**0.463**	**0.689**	**0.757**
Average	**0.492**	**0.809**	**0.435**	**0.699**	**0.759**
Non-hierarchical	**0.519**	**0.828**	**0.463**	**0.726**	**0.776**

Table 9 MLBT with different orders of classification on components

First, we are interested in how the order of classification on components affects the annotation performance. In Table 9, the f-scores of a group of MLBTs are shown with different components selected as the tree root and the rest in the

same order as Table 8 (the standard deviations are omitted and the f-scores of the original order are also listed for comparison).

	Instruction	Stimulus Modality	Stimulus Type	Response Type	Response Modality
ST*>SM>RT>RM>I	0.546	0.856	~~1.000~~	0.708	0.772
SM*>ST>RT>RM>I	0.543	~~1.000~~	0.551	0.732	0.782
RT*>ST>SM>RM>I	0.544	0.817	0.446	~~1.000~~	0.956
RM*>ST>SM>RT>I	0.568	0.818	0.448	0.915	~~1.000~~
I*>ST>SM>RT>RM	~~1.000~~	0.864	0.532	0.844	0.893
Average	**0.550**	**0.839**	**0.494**	**0.800**	**0.851**
Original Average	**0.492**	**0.809**	**0.435**	**0.699**	**0.759**
Non-hierarchical	**0.519**	**0.828**	**0.463**	**0.726**	**0.776**

Table 10 Enhanced MLBT with one component manually annotated

As can be seen in the table, no order outperforms all the others on every component. This indicates the order of classification on the components has no significant impact on the annotation performance. Another fact that can be observed is that the component which is classified first at the root always gives the best annotation performance among all the orders. Note that when the classifier associated with the root is trained on a certain component, the input training examples are the whole set of abstracts, unlike those classifiers trained on lower tree levels only with a subset of the abstracts. This means for the component trained on top the annotation performance is equivalent to that of applying a basic naive Bayes classifier to the component independently, therefore the f-scores of classifying the components one by one in a flat, non-hierarchical manner with naive Bayes can also be known from this figure, summarized in the last row of Table 9. Although MLBT seems not to gain any advantage over the common, non-hierarchical approach, Table 10 demonstrates that an 'enhanced' version of MLBT (MLBT*) with a little adaption can significantly outperform it.

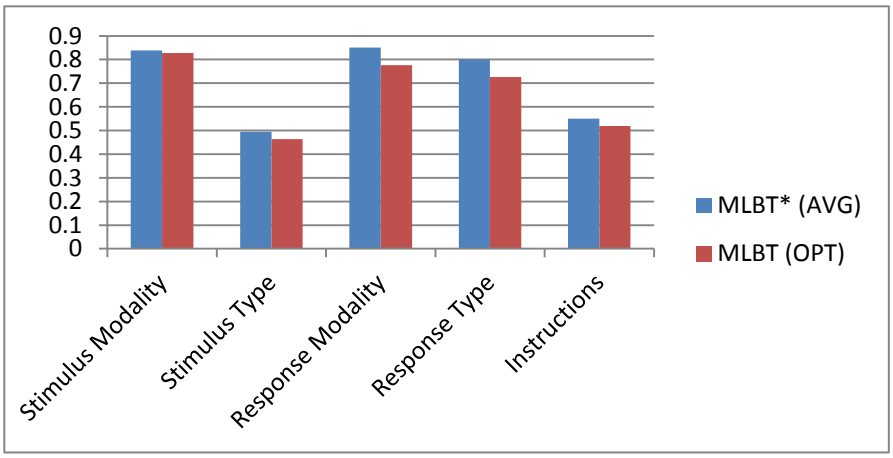


Figure 7 Comparisons between MLBT and enhanced MLBT (MLBT*)

Our enhancement is to call manual annotation at the root in the generalization phase. It means when classifying a new abstract, instead of using a basic naive Bayes classifier to classify an abstract on the component at the root, we pull the 'correct' labels for the corresponding component directly from manual annotation results and feed them into the decision tree. The rest of the process is the same as the original MLBT approach. The new f-scores are shown in Table 10. Since we feed the correct labels directly for the first component (followed with a '*'), the f-scores for the components at the root in each order is exactly 1, which are left out of calculating the average score. As can be seen in Figure 7, MLBT* outperforms both the original MLBT approach as well as the common non-hierarchical way on every component with a clear margin. These results show that in our classification problem it is possible to explore the label dependencies among different components and improve the automatic annotation performance by using the multi-label Bayesian decision tree approach.

Chapter 5

Conclusions and Future Work

Ontologies can serve to organize concepts and structure information in a specific domain. It provides people with insight and exposes the nature of the information to be processed. Combined with conventional machine learning techniques, the effectiveness and efficiency of knowledge processing information can be significantly improved.

We propose a stochastic approach using Multi-label Bayesian Decision Trees which integrates naive Bayes classifiers with decision trees to automatically annotate neuroimaging literatures in biomedical science and the Cognitive Paradigm Ontology. It enables biomedical scientists to find research papers of their interest efficiently, therefore saving them work in manual literature reading and categorization.

Our experiments show that associating the annotation process with the ontology leads to much better performance in automated annotation in comparison with contemporary standard annotators such as the Stanford NCBO annotation tool, which are usually based on pure flat text mining methods. By working in the framework of an ontology, our approach takes advantage of the inherent relationship among concepts in the knowledge domain and narrows down the input sample space to produce better prediction on labels. Although our algorithm cannot completely replace a human annotator, it can effectively reduce the manual effort made on the annotation process by a human expert.

With Brainmap Tracker large-scale identification of studies examining brain activation patterns is possible. In the next step the algorithm developed can be applied to the entirety of abstracts indexed on PubMed, to search and retrieve targeted subsets of studies that are candidates for meta-analysis. Furthermore, by identifying coherent groups of studies suitable for neuroimaging meta-analysis, more versatile tools can be implemented to automate extraction of neuroimaging results, add structured information regarding the experimental methods for better integration and interpretation.

Reference

[1] www.ncbi.nlm.nih.gov/pubmed (2012)

[2] Doms A., Schoeder M.: *GoPubMed: exploring PubMed with the Gene Ontology*. Nucleic Acids Research, vol. 33, Web Server issue (2005).

[3] en.wikipedia.org/wiki/Information_retrieval (4.2.2013)

[4] en.wikipedia.org/wiki/F-score (4.2.2013)

[5] en.wikipedia.org/wiki/Precision_and_recall (4.2.2013)

[6] en.wikipedia.org/wiki/Text_mining (4.2.2013)

[7] en.wikipedia.org/wiki/Pattern_recognition (4.2.2013)

[8] en.wikipedia.org/wiki/Natural_language_processing (4.2.2013)

[9] en.wikipedia.org/wiki/Neuroimaging (4.2.2013)

[10] en.wikipedia.org/wiki/Brain_mapping (4.2.2013)

[11] brainmap.org (4.2.2013)

[12] en.wikipedia.org/wiki/Ontology (4.2.2013)

[13] www.cogpo.org (4.2.2013)

[14] www.bioontology.org (4.2.2013)

[15] en.wikipedia.org/wiki/Machine_learning (4.2.2013)

[16] en.wikipedia.org/wiki/Supervised_learning (4.2.2013)

[17] en.wikipedia.org/wiki/Unsupervised_learning (4.2.2013)

[18] en.wikipedia.org/wiki/Statistical_classification (4.2.2013)

[19] McCallum A. K.: *Multi-label text classification with a mixture model trained by EM.* Proceedings of AAAI' 99 Workshop on Text Learning, pp. 1-7 (1999).

[20] Cortes C., Vapnik V.: *Support Vector Networks*. Machine Learning, 20(3), pp. 273-297 (1995).

[21] en.wikipedia.org/wiki/Naive_Bayes (4.2.2013)

[22] Tsoumakas G., Katakis I.: *Mining Multi-label Data.* Maimon O., Rokasch L.(Ed.), Springer, 2nd edition (2010).

[23] Chen W., Yan J., Zhang B., Chen Z., Yang Q.: *Documentation transformation for multi-label feature selection in text categorization*. Proceedings of 7th IEEE International Conference on Data Mining, Los Alamitos, CA, USA, IEEE Computer Society, pp. 451-456 (2007).

[24] Read J.: *A pruned problem transformation method for multi-label classification*. Proceedings of New Zealand Computer Science Research Student Conference, pp. 143-150 (2008).

[25] Tsoumakas G., Katakis I.: *Multi-label classification: an overview*. International Journal of Data Warehousing and Mining, 3(3), pp. 1-13 (2007).

[26] Fürnkranz J., Hüllermeier E.: *Preference learning*. Künstliche Intelligenz, 19(1), pp. 60-61 (2005).

[27] Brinker K., Fürnkranz J., Hüllermeier E.: *A unified model for multilabel classification and ranking*. Proceedings of the 17th European Conference of Artificial Intelligence, pp. 489-493 (2006).

[28] Fürnkranz J., Hüllermeier E.: *Pairwise preference learning and ranking*. Proceeding of the 14th European Conference on Machine Learning, pp. 145-156 (2003).

[29] Har-Peled S., Roth D., Zimak D.: *Constraint classification: a new approach to multiclass classification and ranking*. Advances in Neural Information Processing Systems 15 (2002).

[30] en.wikipedia.org/wiki/Decision_tree_learning (4.2.2013)

[31] en.wikipedia.org/wiki/Information_entropy (4.2.2013)

[32] en.wikipedia.org/wiki/Information_gain_in_decision_trees (4.2.2013)

[33] en.wikipedia.org/wiki/Overfitting (4.2.2013)

[34] Quinlan J. R.: *Induction of decision trees*. Machine Learning, 1(1), pp. 81-106 (1986).

[35] Quinlan J. R.: *C4.5: programs for machine learning*. Morgan Kaufmann Publishers (1993).

[36] Clare A., King R. D.: *Knowledge discovery in multi-label phenotype data*. Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery, pp.42-53 (2001).

[37] Breiman L.: *Bagging predictors*. Machine Learning, 24(2), pp. 123-140 (1996).

[38] Freund Y., Schapire R. E.: *A decision theoretic generalization of on-line learning and an application to boosting*. Journal of Computer and System Sciences, 55(1), pp. 119-139 (1997).

[39] Schapire R.: *BoosTexter: a boosting-based system for text categorization*. Machine Learning,

39(2/3), pp. 135-168 (2000).

[40] Comite F., Gilleron R., Tommasi M.: *Learning multi-label alternating decision trees from texts and data*. Proceedings of the 3rd International Conference on Machine Learning and Data Mining in Pattern Recognition, pp. 35-49 (2003).

[41] en.wikipedia.org/wiki/Lazy_learning (4.2.2013)

[42] en.wikipedia.org/wiki/K-nearest_neighbor_algorithm (4.2.2013)

[43] Spyromitros E., Tsoumakas G., Vlahavas I.: *An empirical study of lazy multilabel classification algorithms*. Proceedings of 5th Hellenic Conference of on Artificial Intelligence, pp. 401-406 (2008).

[44] Cristianini N., Shawe-Tayler J.: *An introduction to support vector machines: and other kernal-based learning methods*. Cambridge University Press, New York, NY, USA (2004).

[45] Bishop C. M.: *Pattern recognition and machine learning (information science and statistics)*. Springer (2006).

[46] Godbole S., Sarawagi S.: *Discriminative methods for multi-labeled classification*. Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining, pp. 22-30 (2004).

[47] Ting S. L., Ip W. H., Tsang Albert H. C.: *Is Naive Bayes a good classifier for document classification?* International Journal of Software Engineering and Its Publications, 5(3) (2011).

[48] Devi M. I., Rajaram R., Selvakuberan K.: *Generating best features for web page classification*. Webnology, 5(1) (2008).

[49] Do T. D., Hui S. C., Fong Alvis C. M.: *Associative feature selection for text mining*. International Journal of Technology, 12(4) (2006).

[50] Chakrabarti S., Roy S., Soundalgekar M. V.: *Fast and accurate text classification via multiple linear discriminant projection*. The International Journal on Very Large Data Bases, pp. 170-185 (2003).

[51] en.wikipedia.org/wiki/Cross-validation_(statistics) (4.2.2013)

[52] mallet.cs.umass.edu (4.2.2013)

[53] Mitchell T. M: *Machine Learning*, McGraw Hill (1997)

[54] Fletcher T.: *Support Vector Machines Explained*. unpublished (4.2.2013).

[55] Jones T, Charkrabarti C, Xu J., Turner M., Luger G., Laird A., Turner J.: *Modeling Ontology-Based*

Annotation Process using a Stochastic Framework, OHBM (2013)

Appendix A

Medline	Stim Mod	Stim Type	Res Mod	Res Type	Instruction
11823267	Visual	Words	Oral/Facial	Speech	Name
10080553	Auditory, None	None, Words	None, Oral/Facial	None, Speech	Passive/Rest, Recall
11241873	Visual	Words	Oral/Facial	Speech	Recall
11466121	Olfactory	Odor	None	None	Attend
10739412	None	None	None	None	Passive/Rest
11313038	None	None	None	None	Passive/Rest
9699694	Auditory	Letters, Words	None	None	Generate
10327898	None	None	None	None	Passive/Rest
11578663	Visual	Words	None	None	Generate
10227106	Auditory, Visual	Fixation Point, Letters, Words	None	None	Fixate, Generate
12727696	Visual	Shapes	Hand	Button Press	Discriminate, Recall
14638592	Visual	Digits	Hand	Button Press	Detect, Recall
11053229	Visual	Digits	Hand	Button Press	Detect, Recall
10557338	Visual	Letters	Hand	Button Press	Recall
11728837	Visual	Letters	Hand	Button Press	Detect, Recall
15099600	Visual	Shapes	Hand	Button Press	Detect, Recall
11691686	Visual	Digits	Hand	Button Press	Discriminate, Recall
12598724	Visual	Pictures	Hand	Button Press	Attend
12606841	Visual	Asian Characters	None	None	Generate, Repeat
11431233	Visual	Letters	Hand	Button Press	Detect, Recall
12729869	Visual	Letters	Hand	Button Press	Discriminate, Recall
12151286	Visual	Fixation Point, Letters	Hand, None	Button Press, None	Discriminate, Fixate, Recall
10986548	None	None	None	None	Passive/Rest
12714174	None, Tactile	Eye Puffs, None	None	None	Attend, Passive/Rest
12946085	Visual	Letters	Hand	Button Press	Discriminate, Recall
15050867	Auditory	Tones	Hand	Button Press	Recall
15099603	Visual	Abstract Patterns	Hand	Button Press	Detect, Recall
9673996	None, Visual	None, Pictures	Foot, None	None, Point	Discriminate, Passive/Rest
15741464	Visual	Letters	Hand	Button Press	Discriminate
14674880	Visual	Letters	Hand	Button Press	Discriminate
15949653	Visual	Letters	Hand	Button Press	Discriminate
1410086	None	None	None	None	Passive/Rest
11438629	None	None	None	None	Passive/Rest

1402966	None	None	None	None	Passive/Rest
1527602	None	None	None	None	Passive/Rest
8772633	None	None	None	None	Passive/Rest
12547471	None	None	None	None	Passive/Rest
10974961	None	None	None	None	Passive/Rest
11384897	None	None	None	None	Passive/Rest
15921853	None	None	None	None	Passive/Rest
12063157	None	None	None	None	Passive/Rest
14706942	None	None	None	None	Passive/Rest
11986125	None	None	None	None	Passive/Rest
11063978	None	None	None	None	Passive/Rest
12427580	None	None	None	None	Passive/Rest
10327899	None	None	None	None	Passive/Rest
15992522	Visual	Fixation Point, Words	Hand, None	Button Press, None	Discriminate, Fixate
10641577	None	None	None	None	Passive/Rest
11839364	Visual	Symbols	None, Ocular	None, Saccades	Attend, Fixate
15006650	Visual	Digits, Shapes	Hand	Finger Tapping	Discriminate, Recall
15377745	None	None	None	None	Passive/Rest
12150424	Visual	Faces, Fixation Point, Words	Hand, None	Button Press, None	Fixate, Recall
16199829	Visual	Words	Hand	Button Press	Discriminate
15056518	Visual	Digits, Letters	Hand	Button Press	Discriminate
16054343	Visual	Pictures, Words	Hand	Button Press	Discriminate
12566282	Visual	Letters	Hand	Button Press	Discriminate
11595391	Visual	Pictures	Hand	Button Press	Discriminate
12738340	Visual	Words	Hand	Button Press	Discriminate
11231835	Visual	Letters	Hand	Button Press	Discriminate
15570157	Visual	Words	None	None	Discriminate
16237317	Visual	Faces	Hand	Button Press	Discriminate
12727695	None, Visual	None, Words	Hand, None	Button Press, None	Passive/Rest, Recall
14514494	None, Visual	None, Words	Hand, None	Button Press, None	Passive/Rest, Recall
15993859	Auditory, Visual	Faces, Words	Oral/Facial	Speech	Discriminate
11295369	Auditory	Noise, Tones	Hand	Button Press	Discriminate
14550677	Visual	Pictures, Words	None	None	Attend
15013826	Visual	Faces	Hand	Button Press	Discriminate
16076549	Visual	Shapes	Hand	Button Press	Discriminate, Recall
15339824	None, Visual	Letters, None	Hand, None	Button Press, None	Discriminate, Passive/Rest
12513942	Visual	Letters	Hand	Button Press	Discriminate

15691520	Visual	Faces, Fixation Point	Hand, None	Button Press, None	Discriminate, Fixate
15804721	Visual	Abstract Patterns, Faces	Hand	Button Press	Attend, Recall
16503328	Visual	Letters	None	None	Encode
16503328	Visual	Letters	Hand	Button Press	Discriminate
15541071	Visual	Letters	Hand	Button Press	Discriminate, Recall
15866546	Visual	Pictures	None	None	Attend
12195096	Auditory	Letters, Words	Oral/Facial	Speech	Count, Generate, Repeat
12505803	Visual	Pictures	None	None	Attend
15351766	Visual	Faces, Fixation Point	Hand, None	Grasp, None	Discriminate, Fixate
15955496	Visual	Letters	Hand	Button Press	Discriminate
14754778	None, Visual	Digits, None	Hand, None	Button Press, None	Detect, Generate, Passive/Rest
12411216	None, Visual	None, Words	None	None	Generate, Passive/Rest
15325374	Visual	Pictures	Hand	Button Press	Discriminate
16275810	Visual	Fixation Point, Pictures	Hand, None	Button Press, None	Discriminate, Fixate
16275810	Visual	Fixation Point, Pictures	Hand, None	Button Press, None	Discriminate, Fixate
16806312	Visual	Faces	Hand	Button Press	Attend
15135158	Visual	Fixation Point, Letters	Hand, None	Button Press, None	Detect, Fixate
10903406	Visual	Digits, Shapes	Hand	Button Press	Discriminate, Recall
15841676	Visual	Letters	Hand	Button Press	Discriminate, Recall
11926931	Visual	Shapes	None, Ocular	None, Saccades	Attend, Fixate
9862553	Auditory, None	None, Tones	Hand, None	Flexion/Extension, None	Passive/Rest, Recall
15319275	Visual	Faces, Fixation Point	Hand, None	Button Press, None	Discriminate, Fixate
12513941	None, Visual	Faces, None	Hand, None	Button Press, None	Fixate, Recall
15329304	Auditory, None	Letters, None	None	None	Generate, Passive/Rest
16458267	Visual	Fixation Point, Shapes	None, Ocular	None, Saccades	Fixate, Move, Recall
11050021	Visual	Faces	Hand	Button Press	Discriminate
17069771	Visual	Faces	Hand	Button Press	Discriminate
15750588	Visual	Faces	Hand	Button Press	Discriminate
17151834	Visual	Fixation Point, Letters	Hand, None	Button Press, None	Fixate, Recall
16780808	Visual	Faces, Pictures	None	None	Attend
14625454	Visual	Letters	Hand	Button Press	Detect, Recall
17010573	Visual	Letters	Hand, None	Button Press, None	Discriminate, Fixate
17074949	Visual	Digits	Hand	Button Press	Discriminate, Recall
15187809	Visual	Faces, Shapes, Words	Hand	Button Press	Discriminate
17188464	None, Visual	None, Words	Hand, None	Button Press, None	Discriminate, Passive/Rest
16708026	Visual	Film Clip	None	None	Attend
15741465	None, Visual	Letters, None	Oral/Facial	Speech	Generate, Repeat

16616862	Visual	Faces, Pictures	Hand, None	Button Press, None	Attend, Discriminate
16327784	Visual	Faces, Fixation Point	None	None	Attend, Fixate
17321151	Visual	Faces	Hand	Button Press	Discriminate
12900306	Auditory, Visual	Faces, Words	Hand	Button Press	Discriminate
15235232	None, Visual	Film Clip, None	None	None	Attend
17197102	Visual	Fixation Point, Words	Hand, None	Button Press, None	Discriminate, Fixate
17448605	Visual	Letters	Hand	Button Press	Recall
17476364	None, Visual	None, Words	Hand, None	Button Press, None	Passive/Rest, Recall
16814264	Visual	Pictures	Hand	Button Press	Recall
17337340	Visual	Letters, Words	Oral/Facial	Speech	Generate, Repeat
17517680	Visual	Faces, Fixation Point	Hand, None	Button Press, None	Detect, Fixate, Recall
11229981	None	None	None	None	Passive/Rest
17548751	Auditory, Visual	Pictures, Words	Hand	Button Press	Discriminate
10450253	None	None	None	None	Passive/Rest
16458263	Visual	Words	Hand	Button Press	Discriminate
17010993	Visual	Pictures	None	None	Attend
16983390	Visual	Digits	Hand	Button Press	Recall
17197035	Visual	Letters	Hand	Button Press	Move, Recall
17525987	Visual	Fixation Point, Letters, Shapes	Hand, None	Button Press, None	Count, Detect, Discriminate, Passive/Rest, Recall
16674833	None	None	None	None	Passive/Rest
17182108	Auditory	Words	Oral/Facial	Speech	Recall
17012690	Visual	Words	Hand	Button Press	Read
17403973	Visual	Faces	Hand	Grasp	Discriminate
17885606	Visual	Words	Hand	Button Press	Discriminate
17825123	Visual	Words	Hand	Button Press	Discriminate
17588725	Visual	Words	Hand, None	Button Press, None	Move, Passive/Rest
17547582	Visual	Letters, Words	Oral/Facial	Speech	Detect, Discriminate, Generate, Read
17400195	Visual	Faces	Hand	Button Press	Discriminate
16616832	Visual	Shapes, Words	Hand, None	Button Press, None	Detect, Passive/Rest
18076530	Visual	Letters, Shapes	Hand	Button Press	Discriminate
16108017	Auditory	Tones	Hand	Button Press	Detect
18055184	Visual	Fixation Point, Words	Hand, None	Button Press, None	Discriminate, Encode, Fixate
17768265	Visual	Pictures	Hand	Button Press	Discriminate
8790444	None	None	None, Oral/Facial	None, Speech	Passive/Rest, Recall
16199012	Visual	Words	Hand	Button Press	Discriminate

15177789	Visual	Pictures	Hand	Finger Tapping	Recall
11691685	Auditory, None	None, Tones	Hand, None	Button Press, None	Move, Passive/Rest
16585464	Visual	Abstract Patterns	Hand	Button Press	Recall
11431234	Auditory, None, Visual	None, Words	Hand	Button Press, Finger Tapping	Move, Recall
16814525	Visual	Words	Hand	Button Press	Recall
17020747	Visual	Words	Hand	Button Press	Recall
9819069	Auditory	Digits, Words	None, Oral/Facial	None, Speech	Encode, Recall
10195166	Visual	Letters	Oral/Facial	Speech	Generate, Recall
16497485	Visual	Words	Oral/Facial	Speech	Recall
9626713	Visual	Shapes, Words	Hand	Button Press	Discriminate
8988793	Visual	Fixation Point	None	None	Fixate
14683698	None, Visual	None, Words	Hand, None	Button Press, None	Discriminate, Passive/Rest
17988357	Visual	Faces	Hand	Button Press	Discriminate
15500300	Visual	Faces	Hand	Button Press	Discriminate
15184035	None, Visual	None, Words	Hand, None	Button Press, None	Discriminate, Passive/Rest
18063349	Visual	Faces, Shapes, Words	Hand	Button Press	Discriminate
18329671	Visual	Pictures, Words	None	None	Generate
12887982	Visual	Pictures	Hand	Button Press	Discriminate
18669482	Auditory	Tones	Hand	Button Press	Detect
16377154	Visual	Faces, Fixation Point	None	None	Attend, Fixate
18310580	Visual	Faces	Hand	Button Press	Discriminate
16837058	Visual	Words	Hand	Button Press	Recall
14514501	Visual	Shapes	Hand	Finger Tapping	Move
18837865	Visual	Faces, Shapes	Hand	Button Press	Discriminate
14984424	Visual	Pictures	None	None	Attend, Read
17184978	Auditory	Words	None	None	Attend
14990520	Auditory	Noise, Words	Hand, None	Flexion/Extension, None	Attend, Discriminate
17097071	Visual	Faces	Hand	Button Press	Attend
15289277	Visual	Pictures	Hand	Button Press	Discriminate
15225144	Visual	Pictures, Words	None	None	Read
16112653	Visual	Faces, Words	Hand	Button Press	Discriminate
15094461	Visual	Fixation Point, Words	None, Oral/Facial	None, Speech	Fixate, Read
16225562	Visual	Words	Hand	Button Press	Name
18854323	None, Visual	Letters, None	None	None	Generate, Passive/Rest
17618089	Visual	Letters	Hand	Button Press	Detect, Recall
15541070	Visual	Digits	Hand	Button Press	Discriminate, Recall

16837832	Visual	Words	Hand	Button Press	Count
16135630	Visual	Words	Hand	Button Press	Count
15173843	Visual	Digits	Hand, None	Button Press, None	Attend, Detect
16310510	Visual	Letters	Hand, None	Button Press, None	Detect, Discriminate
15246453	None, Visual	Film Clip	Hand, Oral/Facial	Grasp, Speech	Discriminate, Move
16411978	Visual	Letters, Words	Oral/Facial	Speech	Name
18321870	Visual	Words	Hand	Button Press	Recall
18997158	Auditory	Words	Hand	Button Press	Discriminate
18713781	Visual	Film Clip	Hand	Button Press	Detect, Move
9397017	Auditory	Tones	Hand, None	Grasp, None	Move, Passive/Rest
18571627	Visual	Letters	Hand	Button Press	Discriminate, Recall
19603410	Visual	Letters	Hand	Button Press	Recall
19418510	Visual	Letters	Hand	Button Press	Discriminate, Recall
19118321	Visual	Letters	Hand	Button Press, Flexion/Extension	Detect, Recall
17217921	Visual	Faces, Fixation Point	Hand, None	Button Press, None	Discriminate, Fixate
17656073	Visual	Pictures, Words	None	None	Attend, Discriminate
19243925	Visual	Fixation Point, Shapes	None, Ocular	None, Saccades	Fixate, Move
18954477	Visual	Letters, Words	Hand	Button Press	Discriminate
17916330	None, Visual	None, Shapes	None	None	Attend
19176471	Visual	Faces	Hand	Button Press	Recall
19449330	Visual	Abstract Patterns, Shapes	Hand	Button Press	Discriminate
19500088	Visual	Fixation Point, Shapes	Hand, None	Button Press, None	Fixate, Recall
19624392	Visual	Letters	Hand, None	Button Press, None	Attend, Detect
19442494	Visual	Letters	Hand, None	Button Press, None	Discriminate
19594508	Auditory, Visual	Fixation Point, Letters, Words	Hand, None	Button Press, None	Fixate, Recall
17719567	Visual	Pictures	Hand	Button Press	Discriminate
18559283	Visual	Pictures	Hand	Button Press	Discriminate
17949689	Visual	Faces, Fixation Point	Hand, None	Grasp, None	Discriminate, Fixate
18550030	Visual	Faces	Hand	Grasp	Discriminate
17888408	Visual	Fixation Point, Letters, Pictures	Hand	Button Press	Discriminate
17699669	Auditory, Visual	Pictures, Words	Hand	Button Press	Discriminate, Imagine
9430507	Visual	Pictures	None	None	Attend
18586109	Visual	Shapes	Hand	Button Press	Discriminate
18455373	Visual	Pictures, Shapes	Hand, None	Button Press, None	Attend, Detect
18586275	Visual	Letters	Hand, None	Button Press, None	Discriminate
19389870	Visual	Words	Hand, None	Finger Tapping, None	Move, Passive/Rest

19419384	Visual	Fixation Point, Letters	Hand, None	Button Press, None	Discriminate, Fixate
18950748	Visual	Faces, Words	Hand, None	Button Press, None	Attend, Discriminate
19218875	Auditory, Visual	Letters, Tones	Hand, None	Button Press, None	Detect
18706701	Visual	Pictures	Hand, None	Button Press, None	Attend, Discriminate
19176279	Auditory, None	None, Words	None	None	Attend, Passive/Rest
19239982	Auditory, Visual	Letters, Tones	Hand, None	Button Press, None	Detect, Discriminate
19171077	Visual	Digits, Pictures, Words	Hand	Button Press	Discriminate
17585888	Visual	Letters	Hand	Button Press	Detect
9141092	None	None	None	None	Passive/Rest
19446443	Visual	Fixation Point	None	None	Fixate
17601497	Visual	Letters	Hand	Button Press	Discriminate
15691522	None	None	None	None	Passive/Rest
19448846	None	None	None	None	Passive/Rest
18822408	Visual	Faces, Shapes	Hand	Button Press	Detect
20393460	Visual	Digits	Hand	Button Press	Discriminate
12611834	None	None	None	None	Passive/Rest
19261334	Visual	Shapes, Symbols, Words	Hand	Button Press	Discriminate
19346000	Visual	Faces	Hand	Button Press	Recall
19428222	Visual	Faces, Shapes, Words	Hand	Button Press	Discriminate
19218875	Auditory, Visual	Letters, Tones	Hand, None	Button Press, None	Discriminate
18097655	Visual	Digits, Fixation Point, Shapes	Hand	Button Press	Discriminate
18097655	Visual	Digits, Fixation Point, Shapes	Hand	Button Press	Discriminate
21041614	Visual	Faces	Hand	Button Press	Detect
16203952	None, Visual	None, Shapes	Hand, None	Button Press, None	Passive/Rest, Recall
15885507	Auditory	Noise, Tones	Hand, None	Button Press, None	Attend, Discriminate
11999890	None	None	Hand, None	Finger Tapping, None	Move, Passive/Rest
15169688	Visual	Fixation Point, Words	Hand, None	Button Press, None	Encode, Fixate
16199830	Visual	Fixation Point, Words	Hand, None	Button Press, None	Discriminate, Fixate
16199831	Visual	Letters	Hand	Button Press	Discriminate, Imagine, Recall

Appendix B

Stimulus Modality	f-score	Stimulus Type	f-score	Response Modality	f-score	Response Type	f-score	Instructions	f-score
None	0.649 +/-0.180	3D Objects	0.0 +/-0.0	Arm	0.0 +/-0.0	Blink	0.0 +/-0.0	Attend	0.660 +/-0.337
Auditory	0.0397 +/-0.214	Abstract Patterns	0.0 +/-0.0	Facial	0.0 +/-0.0	Breath-Hold	0.0 +/-0.0	Count	0.587 +/-0.369
Visual	0.933 +/-0.023	Acupuncture	0.0 +/-0.0	Foot	0.0 +/-0.0	Button Press	0.826 +/-0.088	Detect	0.573 +/-0.321
Tactile	0.0 +/-0.0	Asian Characters	0.0 +/-0.0	Hand	0.857 +/-0.098	Draw	0.0 +/-0.0	Discriminate	0.696 +/-0.096
Olfactory	0.0 +/-0.0	Braille Dots	0.0 +/-0.0	Leg	0.0 +/-0.0	Drink	0.0 +/-0.0	Encode	0.890 +/-0.244
Gustatory	0.0 +/-0.0	Breathable Gas	0.0 +/-0.0	None	0.546 +/-0.186	Finger Tapping	0.315 +/-0.754	Fixate	0.740 +/-0.386
Interoceptive	0.0 +/-0.0	Chord Sequences	0.0 +/-0.0	Ocular	0.067 +/-0.632	Flexion/Extension	0.0 +/-0.0	Generate	0.706 +/-0.251
		Clicks	0.0 +/-0.0	Oral	0.0 +/-0.0	Grasp	0.309 +/-0.711	Imagine	0.780 +/-1.247
		Digits	0.074 +/-0.311	Pelvis	0.0 +/-0.0	Manipulate	0.0 +/-0.0	Move	0.766 +/-0.291
		Electrical Stimulation	0.0 +/-0.0	Shoulder	0.0 +/-0.0	Micturate	0.0 +/-0.0	Name	0.0 +/-0.0
		Eye Puffs	0.0 +/-0.0	Torso	0.0 +/-0.0	None	0.761 +/-0.115	None	0.0 +/-0.0
		Faces	0.606 +/-0.273			Point	0.0 +/-0.0	Passive/Rest	0.809 +/-0.269

	False Fonts	0.0 +/-0.0		**Saccades**	0.457 +/-0.877	**Read**	0.425 +/-0.809
	Film Clip	0.0 +/-0.0		**Smile**	0.0 +/-0.0	**Recall**	0.820 +/-0.153
	Fixation Point	0.044 +/-0.146		**Speech**	0.392 +/-0.496	**Repeat**	0.571 +/-1.019
	Flashing Checkerboard	0.0 +/-0.0		**Swallow**	0.0 +/-0.0	**Sing**	0.0 +/-0.0
	Food	0.0 +/-0.0		**Whistle**	0.0 +/-0.0	**Smile**	0.0 +/-0.0
	Fractals	0.0 +/-0.0		**Write**	0.0 +/-0.0	**Track**	0.0 +/-0.0
	Heat	0.0 +/-0.0					
	Infrared Laser	0.0 +/-0.0					
	Infusion	0.0 +/-0.0					
	Letters	0.0 +/-0.0					
	Music	0.0 +/-0.0					
	Noise	0.0 +/-0.0					
	None	0.624 +/-0.131					
	Nonverbal Vocal Sounds	0.0 +/-0.0					
	Nonvocal Sounds	0.0 +/-0.0					
	Odor	0.0 +/-0.0					
	Pain	0.0 +/-0.0					
	Pictures	0.141					

		+/-0.375			
	Points of Light	0.0 +/-0.0			
	Pseudowords	0.0 +/-0.0			
	Random Dots	0.0 +/-0.0			
	Reversed Speech	0.0 +/-0.0			
	Shapes	0.177 +/-0.298			
	Syllables	0.0 +/-0.0			
	Symbols	0.0 +/-0.0			
	TMS	0.0 +/-0.0			
	Tactile Stimulation	0.0 +/-0.0			
	Tones	0.057 +/-0.361			
	Vibratory Stimulation	0.0 +/-0.0			
	Words	0.510 +/-0.109			