Predicting Mental Functions from Brain Activations

Madhura Parikh*
Department of Computer Science
University of Texas, Austin
mparikh@cs.utexas.edu

Subhashini Venugopalan*
Department of Computer Science
University of Texas, Austin
vsub@cs.utexas.edu

Abstract

Over the past few decades neuroscientists have studied brain images from EEG/MEG, fMRI and other sources to identify associations between psychological tasks and activity in brain regions [**?**]. Although these studies have led to large amounts of literature and several discoveries of cognitive functions associated with certain brain regions (or networks) the mapping between functions to brain regions and vice-versa still remains largely unclear. For the purposes of this project, we look at enhancing a new automated framework NeuroSynth [**?**] that combines text-mining and machine learning techniques to generate probabilistic mappings from neural activations to cognitive functions. Starting from neurosynth's Naive Bayes classifier, we apply more sophisticated binary classifiers to the problem such as logistic regression, SVMs and ensemble methods. Additionally we also address the problem of multi-label predictions to predict multiple functions for a single image. We use several binary classifiers including clustering techniques to perform *One-vs-All* decomposition. Finally, we perform transfer learning, transferring our learned models from a synthesized domain to the actual fMRI data. Our results are preliminary and encouraging with regard to specific mental functionalities.

1 Introduction

For the purposes of this project, we look at enhancing a new automated framework NeuroSynth [**?**] that combines text-mining and machine learning techniques to generate probabilistic mappings between cognitive and neural states. The framework addresses the following problems:

Forward Inference: In simple terms, given a psychological function or concept, the forward inference(a.k.a encoding) answers what regions of the brain are activated during that function. e.g. pain, vision, conflict, etc

Reverse Inference: The reverse inference(a.k.a. decoding) looks at the reverse mapping. i.e. Given a signature of neural activity, reverse inference identifies the cognitive state(s) and functions that the activations correspond to.

In this project, we primarily address the problem of Reverse Inference Figure 1). The scientific community typically uses fMRI scans for reporting this neural activity. Reverse inference is an extremely challenging problem since multiple cognitive states could have very similar neural signatures [**?**] but it is also of major interest to the neourimaging community at large.

*Authors contributed equally

Figure 1: The reverse inference problem

2 Related Work

Forward and reverse inference problems have been addressed by several contemporary works [**?**, **?**, **?**, **?**]. Previous approaches have generally tackled the Reverse Inference problem by manually analyzing fMRI scans of subjects, collected from the laboratory. There are several limitations of such an approach - for instance, involving human subjects for fMRI scans is labor and cost intensive and the number of data samples that can be gained from such efforts is also very less. Moreover all the meta-analyses based on such data is carried on a very small-scale at individual research labs, and fails to take advantage of the vast knowledge embodied in the entire research community. In this report we will focus on neurosynth since it is most relevant to our approach. For a general discuss of other related works we request the reader to kindly refer to our proposal.

NeuroSynth's (and therefore our) approach is unique - in that we tackle the Reverse Inference problem not by requiring actual fMRI scans, but rather by exploiting the relatively large repository of neuro-imaging publications using text-mining and machine learning techniques. There are many motivations and benefits that lead to this. For one, while fMRI scans are very few, there has been a growing body of publications related to neuro-imaging, thus offering a much larger source of data. Further by using machine learning techniques the decoding is possible without any real training data (fMRI scans) and at the same time incorporates the knowledge base derived from several researcher. Also to the best of our knowledge, this is the first approach that is fully automated, thus making it possible to perform several meta-analyses on a much larger scale than could ever be possible by individual researchers. In the next few paragraphs, we introduce the NeuroSynth framework and some of the techniques it utilizes.

2.1 The NeuroSynth framework

The figure 2 gives a high-level view of NeuroSynth.

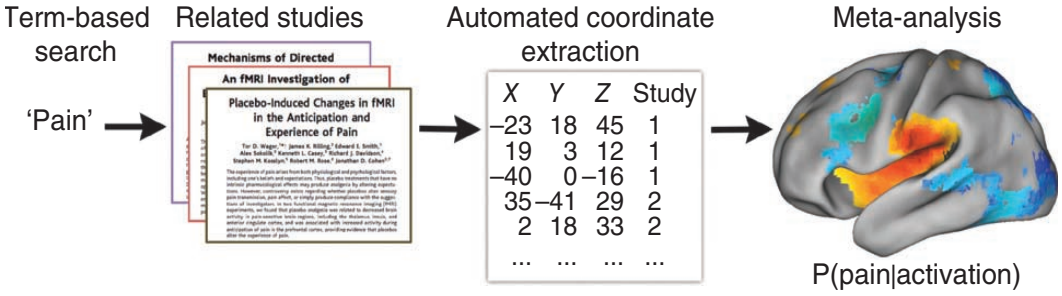


Figure 2: The NeuroSynth framework [**?**]

A detailed description of NeuroSynth may be found in [**?**]. Here we give only a high-level view - figure 2. For reverse inference, NeuroSynth performs the following three steps:

Step 1: Extract high frequency terms First, a database of nearly 3000-odd [1] studies is scrapped to extract the most frequent terms and their frequency of occurence across these studies. Thus corresponding to each study we get a set of terms. These set of terms serve as labels for that study, during the classification.

Step 2: Extract coordinates of activation foci and synthesize sparse image Using simple template matching, all probable activation foci mentioned in a study are extracted. Once we have a list of these coordinates, corresponding to each study, they are used to synthesize extremely sparse brain images with the corresponding brain regions activated. After some preprocessing (which we describe in detail in section 3.4) the vectorized image serves as the feature vector for that study.

Step 3: Use classification to build a predictive model Use classification to train a model, give the labels and the feature vectors from steps (1) and (2). They use an extremely simple approach in which they apply a Naive Bayes classifier to make single-label predictions . They pick up 25 of the most frequent terms arbitarily and train $\binom{25}{2}$ models (i.e one-vs-one) corredponding to every term pair and 10-fold cv to make the predictions.

There have been other approaches that have further built on NeuroSynth. For instance in [**?**], the authors use label decomposition techniques in a one-vs-all setting to predict multiclass labels for the reverse inference problem. They use Support Vector Machines (l_2 regularized), logistic regression and ridge classifier for these tasks, comparing the outcomes based on different criteria like precision, recall and Hamming loss, to show that the multiclass approach is effective for reverse inference. They also propose to use other multiclass approaches and regularization techniques and analyze their performance as future work.

In another similar work [**?**] uses a Generalized Linear Model(GLM) for forward inference. For the reverse inference they use logistic regression with Ward clustering to counter the high dimensionality of the problem.

Building up on these approaches, in the following sections, we shall describe our own extensions and experiments toward building a better model for the reverse inference problem.

2.2 Our Contribution

This work has three primary contributions

1. We improve the single label prediction task by incorporating more sophisticated classifiers such as logistic regression, SVMs and ensembles.
2. We experiment with several classifiers for the multilabel task. In particular we evaluate 6 different models of multilabel classifiers described in greater detail in Section 3
3. Finally we perform transfer learning using one of our better models from the multilabel task. We train our model on the synthesized neurosynth task and test it on a real fMRI database.

3 Methods

Notation : We denote matrices by boldface capital letters $\mathbf{X}$ and vectors by bold-face lower case letters $\mathbf{x}$. The set of real valued D dimensional vectors are denoted by $\mathbb{R}^D$. The set of (target) labels are denoted by the letter $\mathcal{L}$, and $|\mathcal{L}|$ denotes the cardinality of the set of labels.

Our input data consists of brain volume images. We denote by $\mathbf{x}_n$ the n^{th} brain volume with voxels collected in a real valued D dimensional vector. The total number of brain volumes is represented by N. Each brain volume is associated with a set of labels $\mathcal{L}_n = l_1, \ldots, l_K$ chosen from the full set of possible target labels $\mathcal{T} = \cup_{n=1,\ldots,N}\mathcal{L}_n$ with $|\mathcal{L}| = L$.

We approach the task of predicting brain functions as a classification task and address it using the following techniques:

[1] The number has grown to over 8000, since 2011 when [**?**] was published.

3.1 Single Label Prediction

As a first step to measure the feasibility of the task we consider the task of predicting one of two labels for each brain volume. The single label prediction learns a mapping $f : \mathbf{x}_n \rightarrow \mathcal{L}_n$, such that $|\mathcal{L}_n| = 1$ for all n, where $|\mathcal{T}| = |\cup \mathcal{L}_n| = 2$. This essentially learns a simple binary classifier for all pairs of labels. Each classifier is trained and tested on an appropriate subset of the data where the labels for the brain volume are unique. We experimented with the following binary classifiers:

Naive-Bayes We built pairwise naive-bayes classifiers as a baseline to replicate the results presented in [**?**].

Logistic Regression An l_1 regularized logistic regression was our next model of choice for a more powerful classifier.

Linear SVM We use a linear SVM which scales well with the size of our features. The hyperparameter C is optimized by a grid search in the space $C = \{1, 10, 100, 1000\}$.

Ensemble We finally build an ensemble of the above three classifiers and consider the label that is the majority in the output of the above three classifiers.

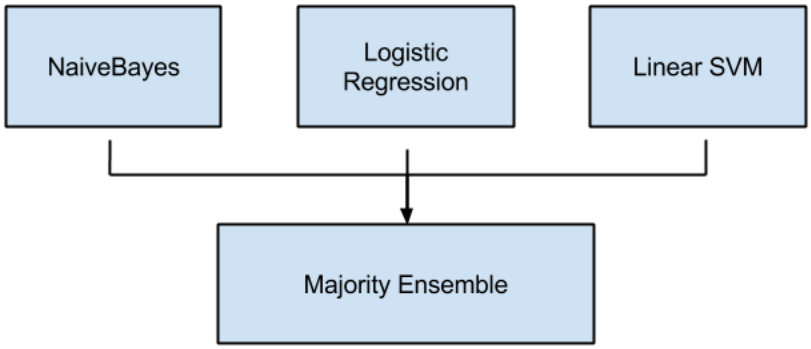


Figure 3: We use naive-bayes, logistic regression, SVMs and a majority ensemble for pairwise binary classification of the labels.

3.2 Multi-Label Prediction

Our next approach is to perform multilabel classification to predict multiple functions for a given brain volume. This learns a mapping $f : \mathbf{x}_n \rightarrow \mathcal{L}_n$, over the full set of labels (where $|\mathcal{L}_n| \geq 1$) and $|\mathcal{T}| = |\cup \mathcal{L}_n|$. [**?**] suggests several methods to perform multilabel classification which include label projection, label ranking and label decomposition. We prefer the label decomposition method for it's ease of interpretation. In particular we choose the *One-vs-All* decomposition approach. This method learns multiple binary classifiers ($|\mathcal{T}|$ to be precise), one for each label to predict the presence/absence of that label. Our experiments used the following binary classifiers for the multi-label prediction task:

Naive-Bayes We built pairwise naive-bayes classifiers as a baseline to replicate the results presented in [**?**].

Logistic Regression We build two models here. The first with l_1 or *lasso* regularization and the other with l_2 or *ridge* regularization.

Linear SVM In a manner similar to our single label prediction, we use a linear SVM by setting the hyperparameter C via a grid search in the space $C = \{1, 10, 100, 1000\}$.

Ward Clustering Since the number of features in our data is extremely large, we do a dimensionality reduction of the features by first performing ward clustering [citation] and then mapping the points to different clusters. The number of clusters is 5000. We then apply a logistic regression classifier with l_1 regularization on the reduced data.

Unique train Since our data is quite sparse and there's a high possibility of noise in data with multiple labels, we build what we call a unique train model based on logistic regression. In the unique train model, we identify all brain volumes with a single unique label, and train

a classifier for each label using only this data. During test, we use the One-vs-All approach to test on multilabel brain volumes.

All ones Another baseline we consider is the all ones baseline, which predicts ones for all labels on all input data points. Such a manipulation will give result in 100% recall for the multilabel task, but very low precision values.

In addition to the above, we did experiment with PCA and k-nearest neighbours for $k \in \{2, 3, 4\}$ but the models were not trained similarly or performed extremely poorly and hence we do not report results for these.

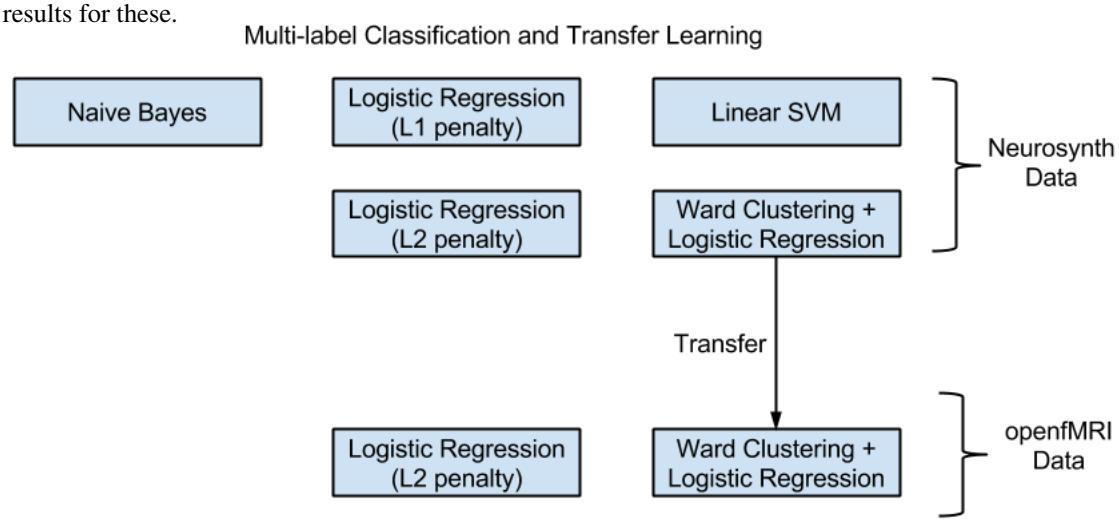


Figure 4: We experiment several classifiers for the multilabel task. We pick our best performing models, namely logistic regression with l_2 penalty and the ward clustering based models and evaluate their performance on the openfMRI data.

3.3 Transfer Learning

As mentioned in Section **??**, one of the main purposes of this particular synthesis method for predicting brain functions is to be able to transfer the models from the synthesis domain to the real openfMRI domain. Based on the mapping functions that we learned in Section 3.2 we want to apply the same function or a small modification of the function to new studies. That is, apply the mapping $f : \mathbf{x}_n \rightarrow \mathcal{L}_n$, to $\mathbf{x}'_n$ where ($\mathbf{x}'_n \notin \mathbf{X}$). In fact, our features $\mathbf{x}'_n$ are different both in dimension and constitution (the elements are continuous real valued data unlike the binary synthesis data). More details about the pre-processing is mentioned in Section **??**. In addition, we are interested in a predicting a different set of labels or functions, as constrained by the availability of data in the new domain. In effect, the transfer learning is attempting to learn $f' : \mathbf{x}'_n \rightarrow \mathcal{L}_n$. We experiment with the following methods here:

Ward Clustering We reuse the ward clustering model that we learn previously by first performing ward clustering [citation] on the data in the synthesized domain to cluster and map the points to different clusters. We then train an l_2 regularized logistic regression *One-vs-All* classifier on the data. Next we reapply the same clustering transformation to the openfMRI data and use the logistic classifier to predict the labels. The number of clusters is 5000.

Logistic Regression For the sake of comparison we train an l_2 regularized logistic regression model directly on the openfMRI data.

Similar to [**?**], we primarily use the repository available with NeuroSynth for our project. This has nearly 8067 studies that are scrapped from online resources, spanning 15 journals. For our transferlearnign approach, we would like to use real fMRI scans, and adapt our models tuned with the synthesized data on this real data. We use data from 2 sources. We use the open source openfMRI [2]

[2] `https://openfmri.org`

project [**?**] to gain the training data. This has individual subject level images for 26 contrasts, resulting in a total of 479 data samples. Since it is always more advisable to work with group level statistics, we also consider the NeuroVault[3] data that has 17 studies, but with group level images. Ideally we would have liked to use this data entirely but the major issues is that it is too less to be meaningful on its own.

3.4 Pre-processing

Here we describe mainly the preprocessing pipe for the NeuroSynth data. The code repository for NeuroSynth already has the labels and coordinates that are extracted from the 8000 studies. Beginning with this, we used various open source toolkits like NumPy tand existing utiliities in NeuroSynth to synthesize images corresponding to the the activated coordinates.

Step 1: First for each (x, y, z) coordinate extracted from a study, we transformed it to the MNI space - which is the standard space used for neuro-imaging.

Step 2: Second, since the coordinates were extracted using simple template matching, there would likely be some spurious numbers that were mistaken as coordinates. It is also possible that the study mentioned coordinates for some different region other than the brain. To deal with such anomalies, we then applied a 2mm MNI mask on the coordinates, to validate that they indeed lay in the brain space and discarded the invalid ones. The 3D mask has the shape $(91x109x91)$

Step 3: Since each study mentions only a very few coordinates from the entre brain space, the resulting image we would get would be extremely sparse. Based on connectivity and correlation amongst the brain regions, given an activated focus, it is highly likely that the voxels in its neighbourhood would be actvated as well. Using this fact we considered all voxels in a 6 mm radius around the activated foci to also be activated and thus get a more dense and smoother image.

Step 4: Once we get this synthesized image in 3D MNI space, we reshape it to a 1D vector, further removing all zero-columns, to end up with a $1x228453$ feature vector.

Since the data is extracted based on simple text processing, there will likely be irrelevant data that is also included. To deal with this we apply some further steps, in which we only consider terms that have a normalized frequency count > 0.001. Further we only consider those studies to be valid, that have atleast 500 activated voxels in the final feature vector. Further for single-label classification we only consider studies, that have a unique label with normalized frequency > 0.001. Studies that have multiple labels at a frequency > 0.001 cannot e clearly assigned a single correct label, and we therefore do not consider them for the single-label classification problem.

For the sake of comparibility, we use the same 22 terms as labels for our experiments as [**?**][4] . After we perform the filtering steps we mentioned above, we end up with a total of 2464 studies out of the original 8067 studies, for the single label classification task.

For the multi-label classification, we perform the same filtering and pre-processing as mentioned above, the only difference being that this time we allow multiple labels per study, instead of removing the studies with conflicting labels. Thus we end up with a total of 5463 studies from the original 8067 studies. Similar to [**?**] we plot a histogram that shows the distribution of terms across studies (fig 5).

For the purposes of transfer learning we use the openfMRI and NeuroVault repositories as a source of real fMRI data. The openfMRI data is also in a similar format to the NeuroSynth data. Thus there is a study associated with an fMRI image and these have already been text-processed previously, to give a frequency count for some 19 terms of interest. One issue is that the labels in NeuroSynth do not overlap the labels that were extracted for openfMRI. Thus we pick out terms tha would be similar to the openfMRI terms from the NeuroSynth database. Now using these new 19 terms, we perform a simialr preprocessing as in the multi-label case to filter out studies that would be reliable

[3] `http://neurovault.org`

[4] The paper actually considers 25 terms however, 3 of these were absent in the version of the data we worked with

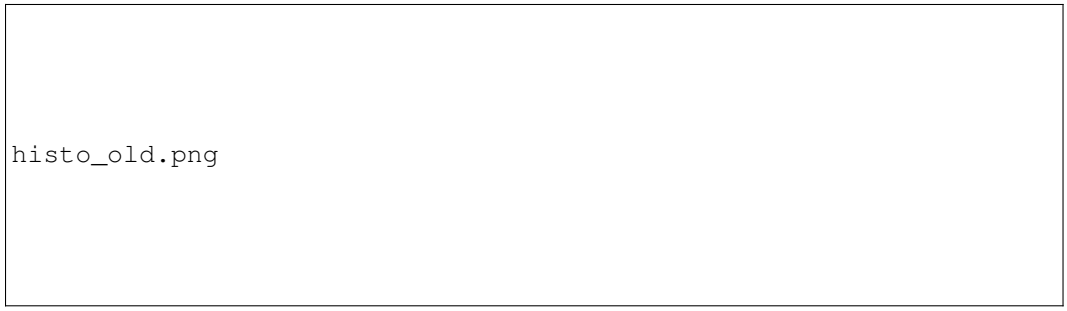


Figure 5: **Distribution of terms across studies:** The x-axis denotes eac term and the y-axis denotes the number of studies in which the term appears.

for these new terms of interest. We end up with 4066 such studies from the NeuroSynth database. Again a distribution of different terms across the studies is presented in fig 6.

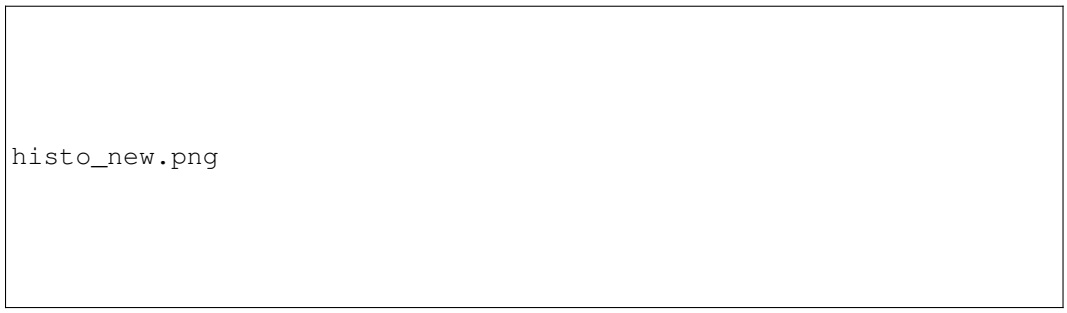


Figure 6: **Distribution of terms across studies:** The x-axis denotes eac term and the y-axis denotes the number of studies in which the term appears.

For NeuroVault, the data was not available in a format similar to the openfMRI and NeuroSynth. In this case, we had a total of 17 images, that corresponded to 12 studies. Rather than having the actual labels for these studies, the API provided us with links to online publications corresponding to the study. These were in the `pdf` format, though in some cases, mutliple formats were also available. We used an approach similar to that already adopted by NeuroSynth, to derive labels for the images. First we converted the `pdf` formats to plain text using an available python tool [5]. Next we considered a dictionary of terms which we were using for the Neurosynth data and counted how frequently these terms appeared in each study. The final frequency we stored for these terms was normalized by dividing it by the total word count of the document. Thus we ended up with a label representation for each study that was analogous to what was being used in NeuroSynth.

4 Experimental Results

Our experiments for single and multilabel prediction tasks were performed on data available in the neurosynth [6] tool. After preprocessing and filtering our inpput feature matrix $\mathbf{X}$ is of dimension $5067 \times 228,453$, the target label matrix $\mathbf{Y}$ is of dimension 5067×23. The transfer learning component is tested on data from the openfMRI [7] project with feature dimensions $\mathbf{X}'$ $(479 \times 174,264)$ and targer label matrix $\mathbf{Y}'$ of dimensions 479×19.

Data samples size (4000). We use 10 fold cross validation to evaluate all our models unless otherwise indicated. We evaluate the performance of our models on the metrics such as *accuracy, precision,*

[5] http://www.unixuser.org/ euske/python/pdfminer/

[6] `www.neurosynth.org`

[7] `https://openfmri.org`

recall, hamming loss, F1 score which are frequently applied to this task.

$$\text{Accuracy} =$$

4.1 Single Label Prediction

We present the accuracies of the pairwise single label predictions using all our approaches in the heat maps below.

4.2 Multi-Label Prediction

In addition to reporting the accuracy, we report precision, recall, F1 score and hamming loss. And one correct

$$\text{Precision} = \qquad \text{Recall} =$$
$$\text{F1 score} = \qquad \text{Hamming loss} =$$
$$\text{Top correct} = \frac{1}{N}\sum_{n=1}^{N}\mathbb{I}(\mathcal{Z}_{n,max} \in \mathcal{L}_n)$$

$\mathbb{I}$ denotes the indicator function and $\mathcal{Z}_{n,max}$ denotes the label predicted with highest probability for the brain volume n. The top correct value is the fraction of the data points on which the model's best predicted label (with maximum probability) was present in the true label set.

This table reports the results of the multilabel classification for the entire label set. Note here that there are 2^{22} possible label sets and accuracy is a fairly hard metric to achieve

Model	Accuracy	Precision	Recall	F1 score	Hamming loss	Top correct
Naive Bayes	1.3	**22.36**	**46.04**	**26.63**	0.207	23.7
Logistic L1	3.2	18.92	14.16	14.53	0.111	27.9
Logistic L2	**4.1**	19.58	14.73	15.22	0.108	**30.0**
Linear SVM	3.7	20.34	16.40	16.24	0.115	22.5
Ward Cluster	3.4	17.53	12.99	13.43	**0.107**	29.8
Unique Train*	0.8	**33.98**	22.09	24.83	0.162	**36.0**
All Ones*	0.0	9.78	**100.0**	17.40	0.902	5.38

Table 1: Multilabel classification full tuple results. All values are reported in percentage (%) except hamming loss. Again, excepting for hamming loss where lower loss is better, for all other metrics higher values are better. *Unique Train and All Ones do not use cross validation.

4.3 Transfer Learning

This table reports the results of the transfer learning for the entire label set. We get 0 accuracy for the full label set and hence do not report the value in the table below.

Model	Precision	Recall	F1 score	Hamming loss	Top correct
Logistic L2	**24.05**	28.04	**21.24**	**0.280**	**31.57**
Ward Cluster*	13.63	**35.35**	16.92	0.416	12.09

Table 2: Multilabel classification full tuple results. All values are reported in percentage (%) except hamming loss. Again, excepting for hamming loss where lower loss is better, for all other metrics higher values are better. * Note that clustering employed actual transfer learning, and does not employ cross validation.

5 Discussion

– Why bad results? Or good results?

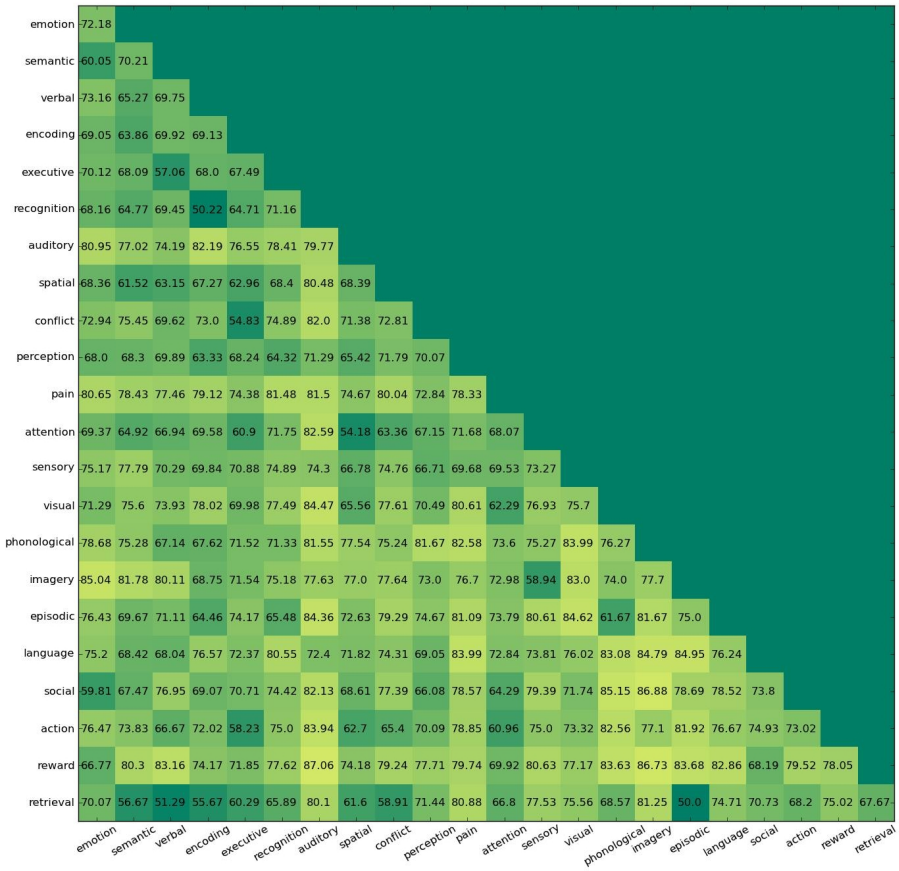
Transfer - remove those studies where none of the terms are present

6 Conclusion

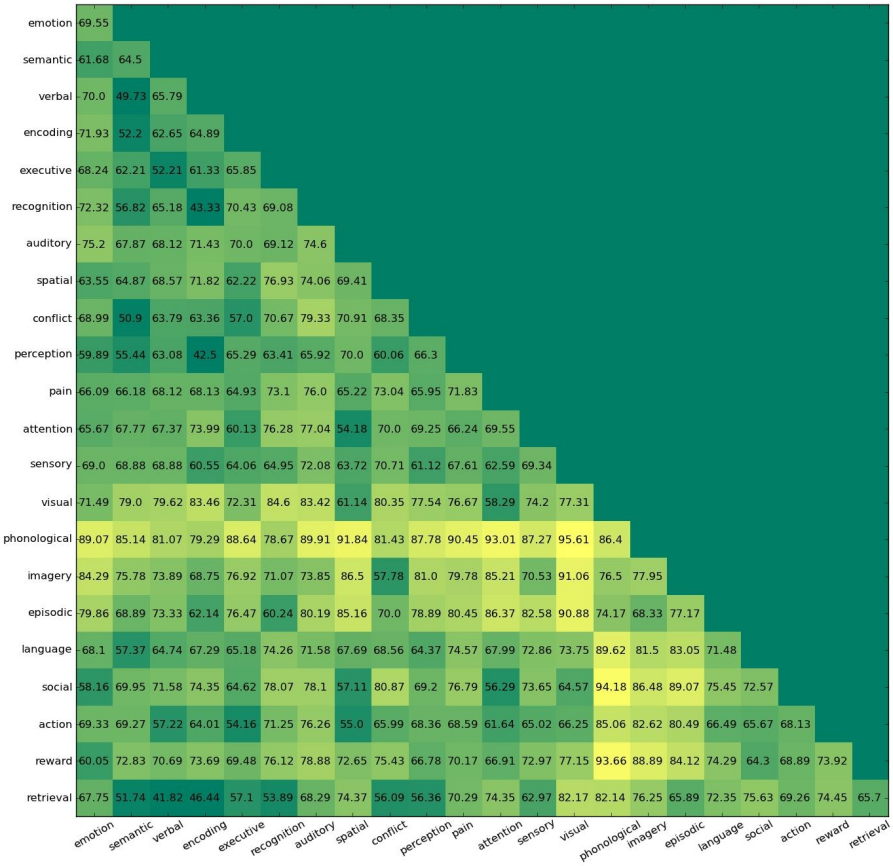
Acknowledgments

Sanmi, Russ, tal, tacc

References

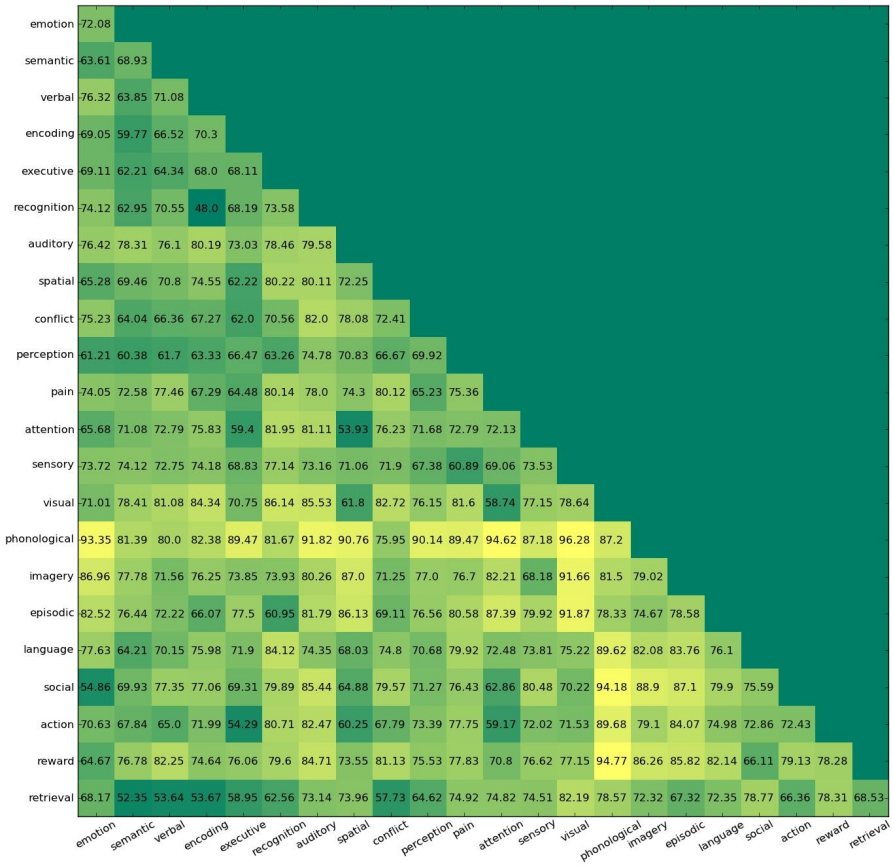


(a) Naive Bayes

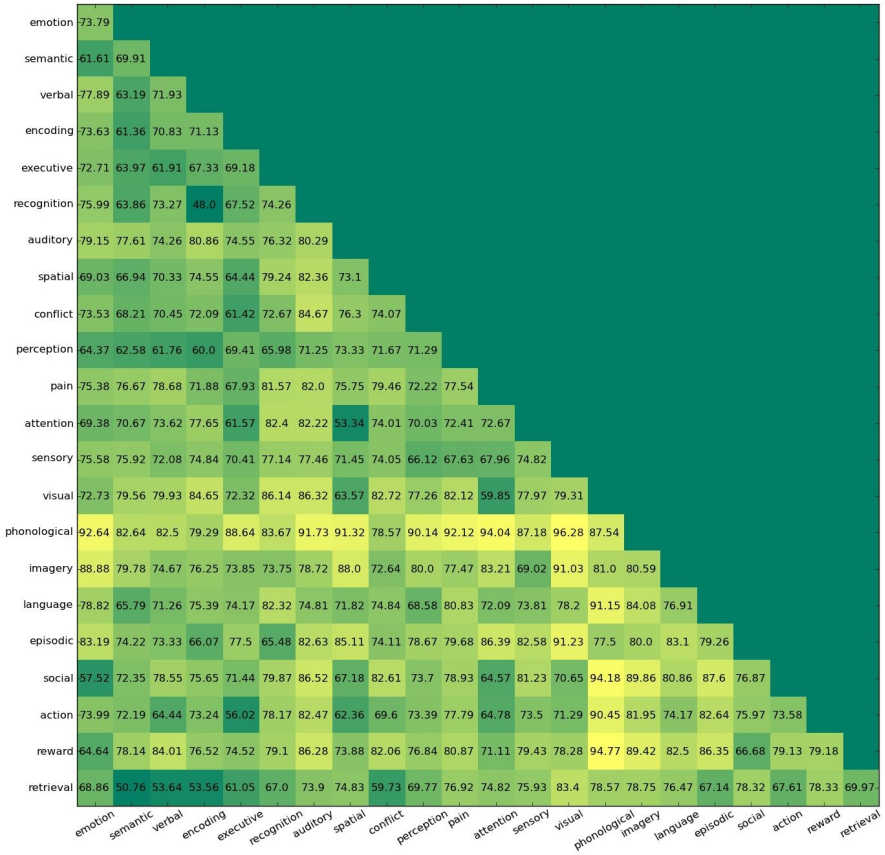


(b) Logistic Regression

Figure 7: Pairwise binary classification accuracy for single label prediction



(a) Linear SVM



(b) Ensemble of naive bayes, logistic regression and SVM

Figure 8: Pairwise binary classification accuracy for single label prediction