Learning to Identify POS from Brain Image Data*

Arshit Gupta
Electrical and Computer Engineering,
Carnegie Mellon University,
Pittsburgh, PA - 15213
arshitg@andrew.cmu.edu

Tom Mitchell
Machine Learning Department,
Carnegie Mellon University,
Pittsburgh, PA - 15213
tom.mitchell@cs.cmu.edu

Abstract

We present a method to decode the parts of speech (POS) in a sentence by observing the brain activity via Magnetoencephalography (MEG). **Naïve Bayes** classifier is used to predict the part of speech from the MEG data collected from multiple sensors (306) of the brain. Further, we explore correlation and cross-correlation between various sensors and attempt to use this correlation to improve the accuracy. Our experiments reveal that we can confidently identify the POS from a given sentence even when we have relatively small dataset. Additionally, we identify brain regions that contribute to correct identification of a particular POS.

1 Introduction

The question of how the brain encodes the conceptual knowledge and thoughts has always been both intriguing and challenging for humans. How the brain encodes the words/pictures we read/see? New technologies in brain imaging along with the usage of numerous machine learning algorithms have made it possible to *understand* how the brain works (although quite vaguely, at the moment). Researchers are using either the fMRI or the MEG data to study the brain activity. We have used MEG in our experiments as it provides high temporal resolution.

Our model predicts the part of speech for each word in a sentence. Our classifier (Naïve Bayes) has five output classes - *Determiner*, *Verb*, *Subject*, *Object* and *Preposition*. Our model not only classifies the words into different parts of speech, but also highlights the regions of the brain that contribute towards a particular part of speech i.e. sensor accuracy w.r.t. time. Our model also tells at what time a particular word is decoded. This lays the foundation for smartly reducing the feature bucket for large datasets to make the algorithm computationally efficient.

Functional connectivity[1], which is the dependency among two or more time-series, provides the insights between the various sensors, which could help in better feature selection. We get time-series data from each sensor when a user reads a sentence (explained in subsequent sections) where in we use various functional connectivity methods such as correlations and cross-correlations of the time series data. Functional connectivity helps in removing the redundant features when we have huge datasets so as to fasten the classification task. Notice that cross-correlations play a crucial role in determining the connectivity among various brain regions.

2 Related Work

As mentioned above, decoding brain data using machine learning techniques has gained a lot of popularity of the past few years. [2] decoded the word a subject reads using MEG data. Our approach is different in a sense that we are not just predicting any random words but the words

*Independent Research, Brain Image Analysis Group - Final Report (Spring 2015)

that occur in the sentence, indirectly inculcating the relationships between the different words in a sentence. [3] did quite similar to what we are trying to do here but with opposite goal - predicting brain activity from nouns. In addition, [3] used fMRI data instead. [4] provides some nice intuition of how a brain activity is a function of various semantic, syntactic, visual and discourse features, that provide great insights which result analysis. [5] used correlation (Canonical Correlation Analysis) to learn fMRI image from a given word (opposite to what we have attempted in this paper)

3 Learning Algorithms

In these preliminary analysis, we are classifying the words of a sentence into different parts of speech (5 in total). We have used Gaussian Naïve Bayes algorithm to perform this classification task:

- **Gaussian Naïve Bayes** - In this model, we assume that each feature is conditionally independent from each other and use this assumption to build a table of conditional probabilities of the observed features given the observed value of the target variable. These probabilities are then used to infer the most likely value of the target variable given the observed features in the test data. While training the samples, we have computed separate means for each class but common standard deviation for all the class as we don't have a huge dataset.

$$\hat{y} = \underset{k \in \{1,\dots,K\}}{\operatorname{argmax}} \; p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{1}$$

$$where, \quad p(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}} \, e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}} \tag{2}$$

  It should be noted here, that log plays an important role while designing the classifier. Since, we have large number of features (306 x 268, explained later), we'll get either zero (if individual features are < 1) or infinity (if individual features are of exponential order) in the numerator if we don't have log.
  Another important point to note here is that for Naïve Bayes, generally mean and variance are computed for every class (for all features) separately. However, in our case, since we have smaller data set, we have calculated a single variance parameter for all the classes.

Gaussian Naïve Bayes (or simply, *Naïve Bayes*, as we'll refer throughout our paper) is widely used for classification tasks, providing quick baseline results.

Additionally, it is important that we define correlation and cross-correlation. *Correlation* is the statistical relationship between two random variables and is defined as:

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

Cross-correlation[6] is the measure of similarity of two time series a function of the lag of one relative to the other.

$$\rho_{xy}(m) = \mathrm{E}[(X_n - \mu_X)\,(Y_{n+m} - \mu_Y)]/(\sigma_X \sigma_Y)$$

where, μ_X and σ_X are mean and standard deviation of the Random variable, X, and μ_Y and σ_Y are mean and standard deviation of the Random variable, Y

4 Experiments

The tool used for the experiment is **MATLAB** (R2013b) and the experiments were run on *Machine Learning Dept. (Carnegie Mellon)* clusters.

4.1 Dataset

We used **KRNS-2-MEG-PassAct** data for our experiment.
Subjects looked a simple 'non-verb-noun' sentences that were either active or passive voicing. Each

word was displayed on screen for finite amount of time (300 ms). We had a total of 480 sentences - 16 (each) unique active and passive sentences, each sentence is repeated 15 times. It included the sentences like *A dog found the peach*, *The peach was found by a dog*, etc. Each word was presented on screen for 300 ms followed by 200 ms resting period. Also, each active sentence was followed by 2000 ms of rest and each passive sentence was followed by 1000 ms of rest. Hence the total duration of each sentence (active/passive) was 4.5 seconds.

The data relevant to our experiment is stored mainly in the three matrices - *time*, *labels* and *data*.

- The *time* matrix is 1 x timeSlots, where each column corresponds to time in milli seconds with a difference of 2ms between the adjacent columns.
- The *labels* is 4 x instances matrix, where instances is the total number of examples/words (2880 in our case). Here, the first row indicates the word id (1-16, corresponding to each unique word in our dataset); the second row indicates whether the sentence, with which the given word is associated, is active/passive; the third row is the sentence id (1-16 for both active/passive); and the forth row is the position of word within the sentence.
- The *data* matrix is a 3D matrix which is sensors x instances x time. This is the main matrix with all the features in it. The sensors are the various locations in the brain contributing to the brain image data. Here, we have 306 sensors.

Note - There is some fixation period for each word presentation which has to be taken into account while preprocessing the data.

4.1.1 Pre-processing

Although most of the preprocessing was already done for us, still some preprocessing was need to be performed. First, the labels did not contain the part of speech. So, we have to manually assign the part of speech to our dataset. For this, we used first and second rows of *labels* matrix to write a script to automatically assign the POS to each word. Second, the *data* had some *corrupt* values corresponding to negative 'time slots'/'fixation periods' (as mentioned in above section). So, those values were removed. Finally, to ease the calculations, *data* matrix was reordered in the following form - instances x sensors x time.

After performing all the preprocessing steps mentioned above, we are left with the parameter values as mentioned below:

- Instances - **2880**
- Sensors - **306**
- Time - **268**
- Output Classes:
  - Class 1 - **Determiner**
  - Class 2 - **Verb**
  - Class 3 - **Subject**
  - Class 4 - **Object**
  - Class 5 - **Preposition**

Therefore, corresponding to each instance, we have 306 x 268 feature bucket. Also, throughout this paper, we'll refer the various POS's as Class 1, Class 2 and so on.

4.2 Results

Here, we will present two approaches of the our goal:

- *Approach I* : Using the given MEG features
- *Approach II* : Using correlation and cross-correlation features along with the given MEG features

Before we present detailed outcome, we would like to re-iterate the point that since we have 5 output classes, *random guessing* will give us 20% accuracy. So, anything above 20% should be considered a positive result.

4.2.1 Approach I

In this approach, we used a 306 x 268 feature bucket. First, we took training data of 2600 and remaining of the instances as our test data (280). After applying Naïve Bayes, we got about 40.00% accuracy which is better than random guessing. Since, our dataset is not huge, we instead decided to perform cross validation to get more accurate results.

Case I: CV - Leave 1 out
Note: The training instances and testing instances are not averaged in this case.
In this case, we iterate through all the examples in the dataset, taking one example as test instance at a time and remaining 2879 as training examples. In this case we get an accuracy of about 49.58% and the corresponding confusion matrix is shown below:

$$ConfusionMatrixI = \begin{bmatrix} \mathbf{489} & 110 & 109 & 108 & 144 \\ 88 & \mathbf{281} & 139 & 93 & 119 \\ 61 & 51 & \mathbf{292} & 48 & 28 \\ 66 & 44 & 77 & \mathbf{275} & 18 \\ 31 & 51 & 34 & 33 & \mathbf{91} \end{bmatrix}$$

where, vertical axis is actual class and horizontal axis represents predicted class. For instance, 5^{th} entry in 1^{st} row indicates that 144 entries, which actually belonged to Class 1, were misclassified as Class 5.

Case II: CV - Leave 2 out
In this case, we take 2878 as our training set and the remaining two examples (which should be the same each time) are averaged. Then, we perform prediction task as before on this averaged example using Naïve Bayes. The advantage of averaging is that it improves the *quality* of the test data. It is evident from output too. We get the accuracy of 61.11% with confusion matrix as shown below:

$$ConfusionMatrixII = \begin{bmatrix} \mathbf{307} & 43 & 35 & 41 & 54 \\ 37 & \mathbf{203} & 51 & 26 & 43 \\ 27 & 28 & \mathbf{162} & 14 & 9 \\ 30 & 19 & 31 & \mathbf{156} & 4 \\ 22 & 30 & 9 & 7 & \mathbf{52} \end{bmatrix}$$

Clearly, we have lower number of misclassifications here.

Case III: CV - Leave 12 out
We continued with increasing the number of examples averaged. We took average of 4, 6 and 12. Accuracy increases in each of the case (Figure 1). It is interesting to note that in the case, where we average 12 words to get a single test data, we get accuracy as high as **88.75%**. These results are quite promising. The confusion matrix for this case is shown below:

$$ConfusionMatrixIII = \begin{bmatrix} \mathbf{78} & 2 & 0 & 0 & 0 \\ 4 & \mathbf{53} & 2 & 0 & 1 \\ 1 & 2 & \mathbf{37} & 0 & 0 \\ 2 & 4 & 2 & \mathbf{32} & 0 \\ 1 & 6 & 0 & 0 & \mathbf{13} \end{bmatrix}$$

All the non-diagonal elements of the matrix are either 0 or very low values. This confirms the fact that Naïve Bayes classifier gives high accuracy when test sample is of *higher* quality.

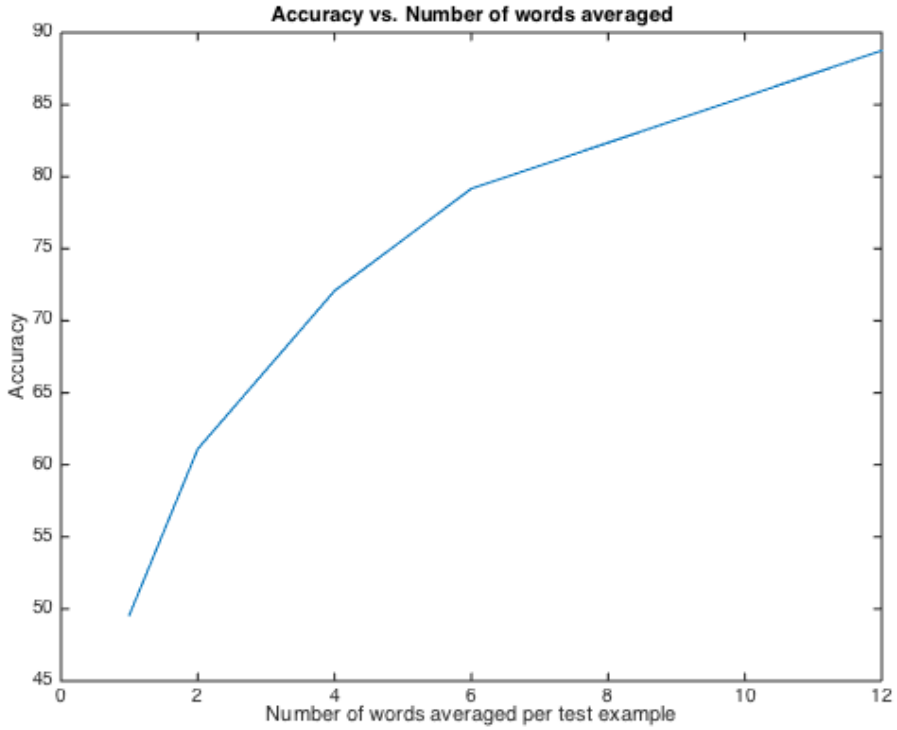

Figure 1: Accuracy vs. Number of words averaged per test sample

4.2.2 Approach II

We feel that the number of features provided by MEG dataset are less which could be hindering the accurately classification of the word. So, we decided to increase the dataset using the correlation and cross correlation relationships among the sensors. The timing window length used in our experiments is 200 ms (**100 time slots**). First, we'll describe the correlation results and then the cross-correlation results

Correlation: Initially, we created a correlation matrix for each of the instance/datapoint. There would be one correlation matrix of size 306x306 for time slots 1-100, showing the correlations among the 306 sensors, and second correlation matrix of size 306x306 for time slots 101-200. Before we began our experiment, we decided to perform sanity check to ascertain the validity of our approach. We used Naïve Bayes algorithm for just these 306x306 generated correlation features and attained the accuracy of 53.44% (leave 12 out - CV accuracy) which is much higher than random chance of 20%.

First implementation was the addition of these new features to the already available MEG features. So, even if we add two correlation matrices to a given instance, we get 306x(268+306+306) i.e. 306x880 matrix for each instance. So the size of the dataset becomes 2880x306x880 (even after ignoring the remainder 268-200 = 68 time slots). MATLAB throws *Out of memory* error (dynamic memory requirements exceed system capacity) while performing functions like calculating variance and mean for such a huge matrix. So, we, instead, decided to add the correlation features for only the first time window (1-100) to the given MEG data. The maximum accuracy that we were about to achieve with this approach was **85.83%** (leave 12 out - CV accuracy) which is less than original accuracy of 88.75%.

Second implementation for the correlation analysis was to *enhance* the given features using the correlation results i.e. for a particular time window, say 1-100, we use correlations among sensors to **average** the corresponding sensors' time-series, e.g. if sensor 1 and sensor 5 are most strongly

related (highest correlation coefficient), then we average time series of both sensors 1 and 5 to improve the *quality* of the features (similar to averaging of 12 instances to improve the quality of testing instance). But, in this implementation, we were able to achieve the accuracy of **88.75%** which is exactly same as original accuracy.

It is worth mentioning that other techniques like usage of binary values only correlation matrix, weighing the MEG features and correlation features, etc were tried but no improvement was observed.

Cross-correlation: Many times correlation between different voxels in brain is separated by time lag i.e. two sensors might same response to a stimulus but one of those sensor's response could be delayed w.r.t. other. In this situation, it makes more sense to use cross-correlation analysis. The cross correlation graphs for sensor 1 with all other sensors (2 - 306) for an arbitrary selected instance from the dataset for a time window of 100 slots (1-100), are shown below:

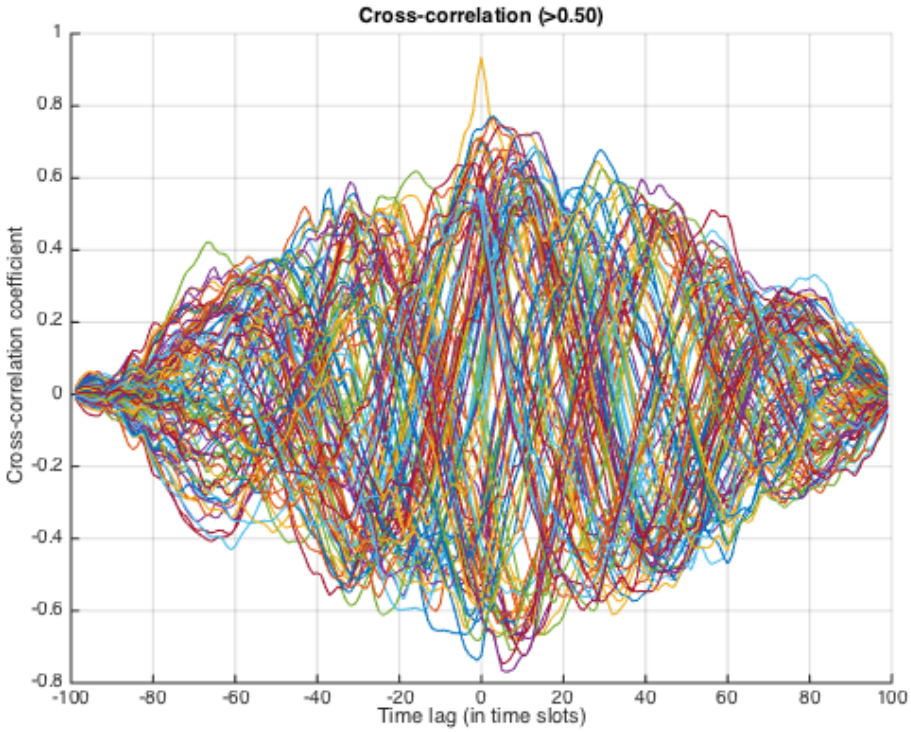

Figure 2: *Cross Correlation* (> 0.5)

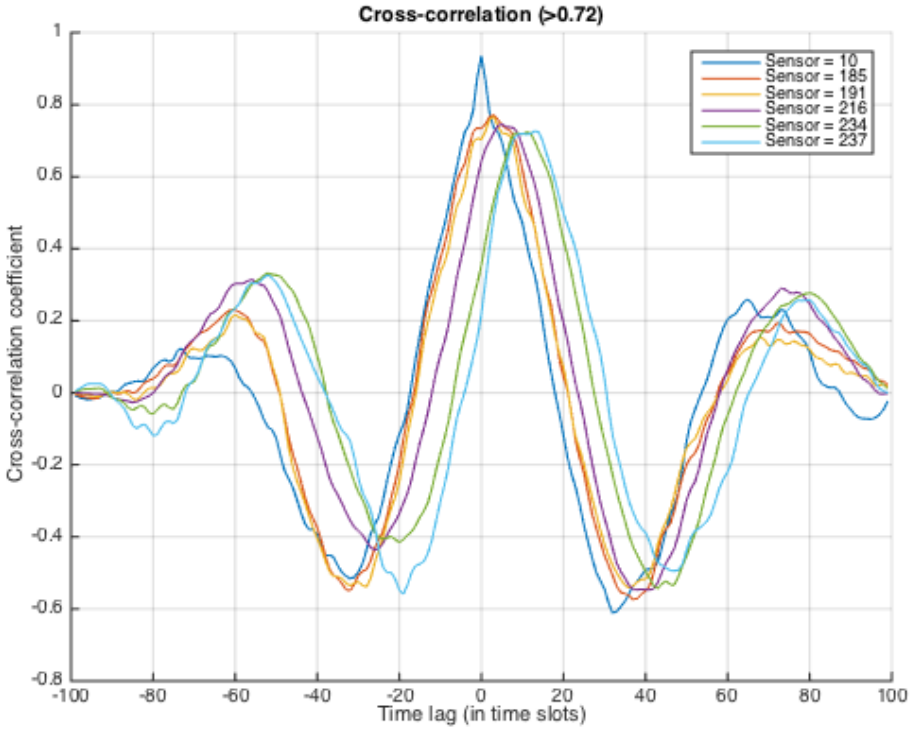

Figure 3: *Cross Correlation* (> 0.72)

Figure 2 shows cross-correlations b/w sensor 1 and all other sensors for whom its value is greater than 0.5. There are large number of such sensors. Figure 3 presents better understanding of cross-correlation. Here, we set cross-correlation threshold as 0.72 so as to have few curves. Sensor 10 has maximum correlation with the given sensor (sensor 1) and it occurs at zero time lag. But situation is not the same for all other sensors. Maximum cross-correlation coefficient (0.725) for sensor 237 with sensor 1 occurs at about time lag of 10 slots (or 20 ms). Hence, cross-correlation analysis becomes important.

Similar analyses were performed for cross-correlation case as for correlation case. Only difference is how we generate cross-correlation matrix for former case. For generating each element/feature of cross-correlation matrix, we take a sensor and find the sensor that gives maximum cross-correlation w.r.t. given sensor. e.g. in the Figure 3, sensor 1 has maximum cross-correlation with sensor 10, so the entry (1,10) of the matrix would be the 0.92.

Surprisingly, we obtain almost similar results when we perform the same implementations for this case. The cross-correlation features only accuracy is 54.5%, cross-correlation features + original features give **85.83%** accuracy, and finally averaging sensor time series values based upon cross-correlation provides the maximum accuracy of **88.75%** which is again identical to the original accuracy. Hence, at least for our dataset, addition of correlation/cross-correlation features doesn't seem to increase the original accuracy.

4.3 Analysis

Is there anything interesting that we can learn from our experiments apart from predicting the POS? Well, the answer is *yes*. During our experiment, we added few lines of code to check whether an individual feature of a test example is consistent with the actual class of that test example. We then added all those features to get an idea as to how strong their contribution is towards the correct prediction of a test example.
Now, we will dwell into the details to give a better perspective. For each test sample, we first apply

Naïve Bayes as shown in equation 1, then we obtain 306 x 268 feature matrix for each class. Hence we have 5 copies of 306 x 268 feature matrix corresponding to each class (5 x 306 x 268). Then we compare the corresponding features of all classes. Indices of features with highest value are stored in separate matrix, *assignMat* (306 x 268). e.g. say feature (1 x 2 x 2) is highest among (1 x 2 x 2), (2 x 2 x 2), (3 x 2 x 2), (4 x 2 x 2) and (5 x 2 x 2). So now (2 x 2) location of the *assignMat* will have 1 as its entry and so on. So, the *assignMat* has values ranging from 1 to 5. Now, since we know the true label of the test sample, we'll keep only those indices of *assignMat* that correspond to the true label of that test sample, and zero the remainder of the elements. This process is repeated until all the samples have been tested (as in the case of Cross Validation) and the corresponding values of these *assignMat*'s are added successively. In this way, we get all those features that contribute to the accurate prediction of the given class (part of speech).

Figures 4 through 8 show that different POS decoded by different sensors in the brain at different times (All these results correspond to Case III: Leave 12 out).

1. **Determiner**: First of all, all the 306 sensors are activated when a determiner is read. It is interesting to note that the real contribution is only at 120 time tick (or 240 ms) mark. The explanation is quite intuitive. In our dataset, most of the determiners occur at the beginning of the sentence. Our brain takes some time to begin to understand the new sentence as there is no prior context for the sentence being read (i.e. all sentences are read independently). This explains as to why the learning occurs at around 240 ms mark.
2. **Verb**: Here, the brain takes mostly first half of the time frame to decode the verb. As you can see in Figure 5, most of the sensors between 40 and 180 contribute towards the correct classification of verb.
3. **Subject**: In this case, learning takes places almost over the entire time frame. It is interesting to note that sensors 1 to 120 and 200 to 300 are the largest contributors towards *subject* classification (75.95%). In other words, sensors between 120 and 300 are quite unresponsive and thus, can be omitted.
4. **Object**: In *object* classification, most of the learning takes place during the end of the time frame. In fact, between 250 ms and 500 ms, the sensors contribute about 56.29%.
5. **Preposition**: Most of the *preposition* identification occurs at the beginning of the time frame which again is quite intuitive. (Figure 8)

Figure 9 and Figure 10 show correlation and cross-correlation respectively, among the sensors from the randomly selected sample. Clearly, diagonal elements in both these figures have the highest value (equal to 1). There are two interesting observations here. First- some regions show high correlation/cross-correlation around the diagonal region. This is be due to the fact that those sensors (for instance, sensors 200 to 250 in Figure 9) are located physically close to each other (hence, the same response). Second- a more interesting case where sensors apart from each other too show strong correlation/cross-correlation (sensor 50 and sensor 210 in Figure 9).

5 Conclusion and Future Work

We have been successfully able to label each word of the sentence with corresponding part of speech (with 88.75% accuracy). The reason for not achieving the accuracy in high 90's could be attributed to the several facts - smaller dataset, using Naïve Bayes classifier which makes conditional independence assumption, etc. Apart from classifying the words in POS, we have been able to show which parts of the brain (sensors) decode various POS and at what accuracy. This could be an asset in reducing the number of features and hence, increasing the classifier's efficiency.

Also, we have tried multiple combinations and permutations of functional connectivities to increase the accuracy. Although many correlations/cross-correlations among various sensors are apparent from our experiments, this approach has not worked as expected. One possible explanation good be that the dataset is too small to add new features to it (overfitting). Hence, higher cross-validation error. Other explanation could be that some "better" classifier is needed that could incorporate both the original features and newly generated features with their respective weights. The next stage of

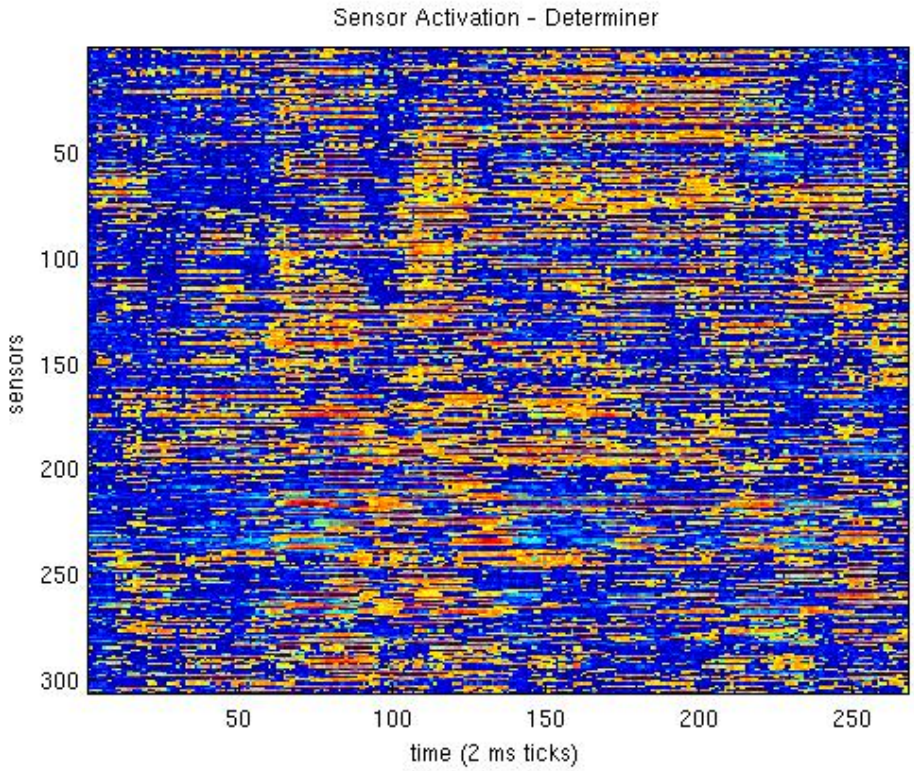

Figure 4: *Sensor Accuracy - Determiner (Class 1)*

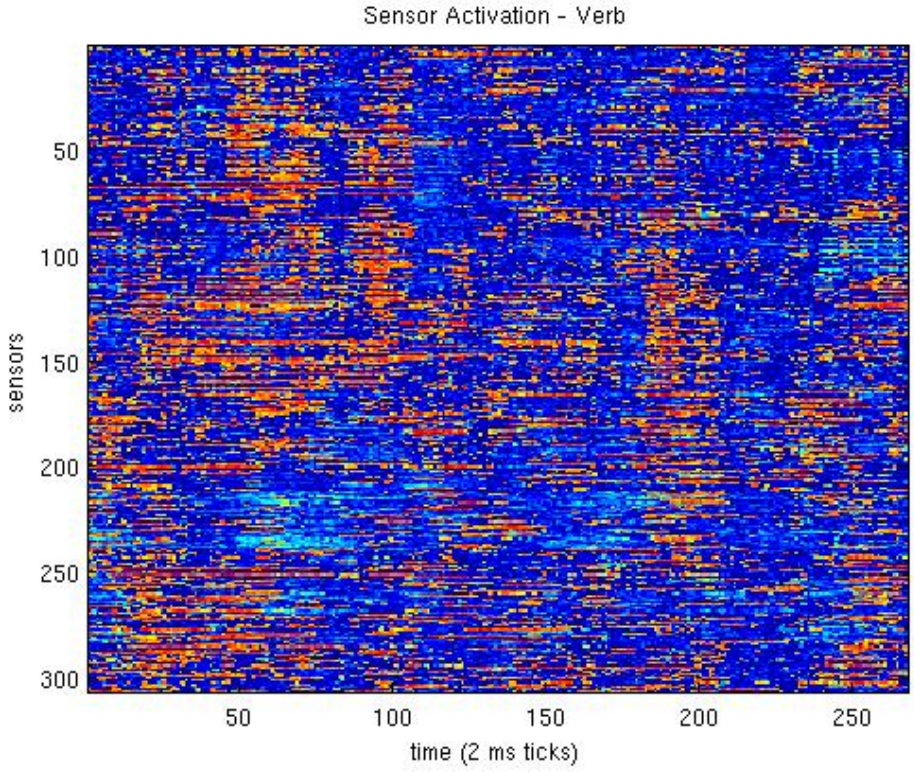

Figure 5: *Sensor Accuracy - Verb (Class 2)*

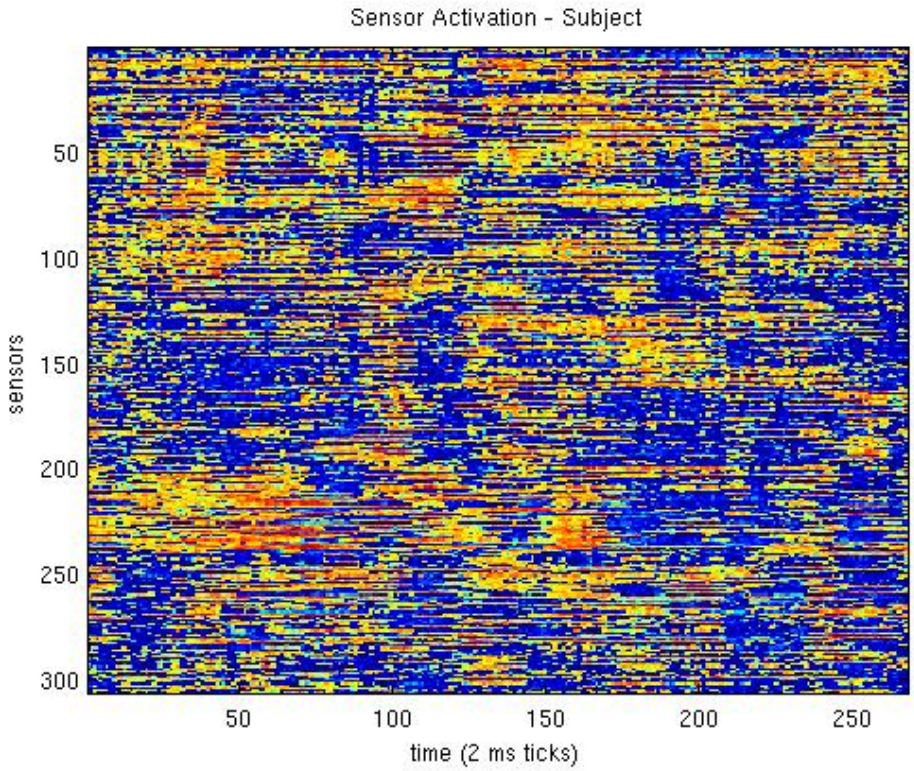

Figure 6: *Sensor Accuracy - Subject (Class 3)*

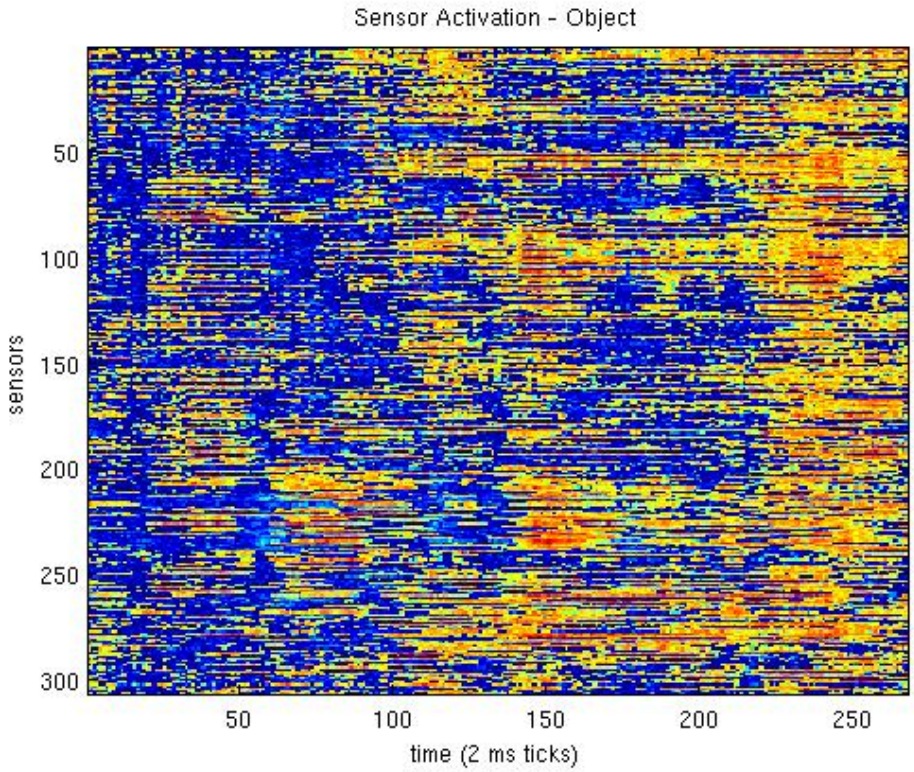

Figure 7: *Sensor Accuracy - Object (Class 4)*

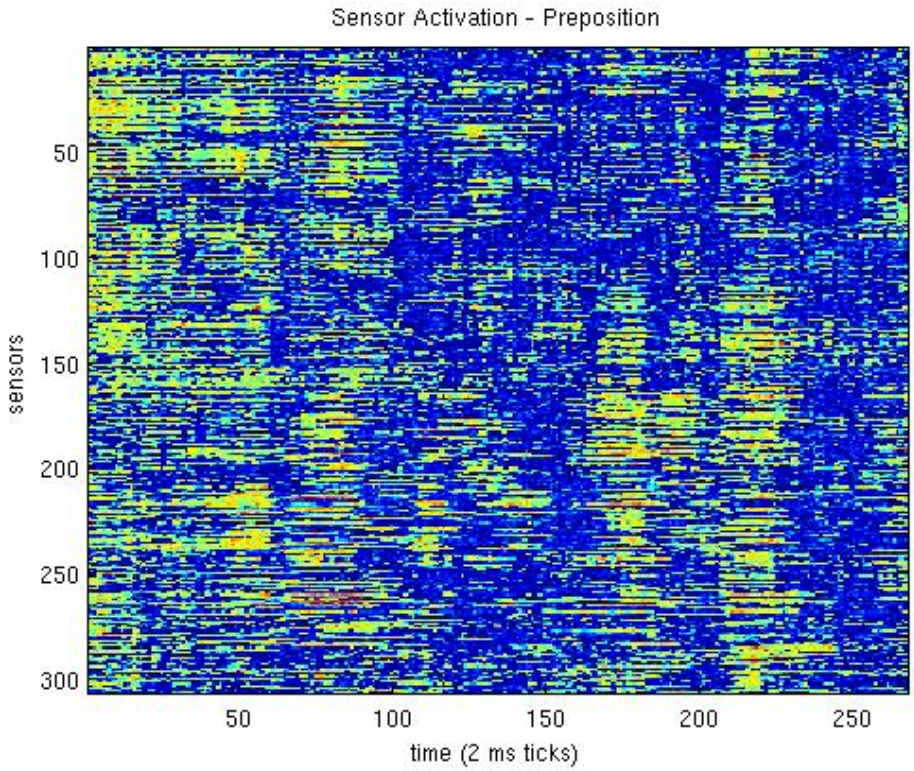

Figure 8: *Sensor Accuracy - Preposition (Class 5)*

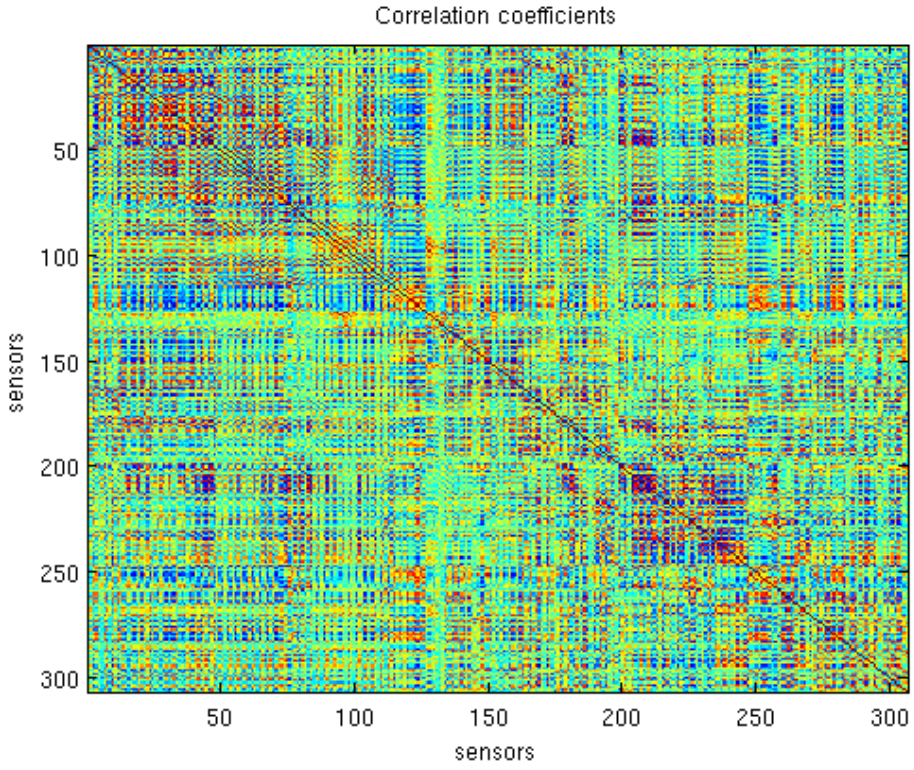

Figure 9: Correlation among features

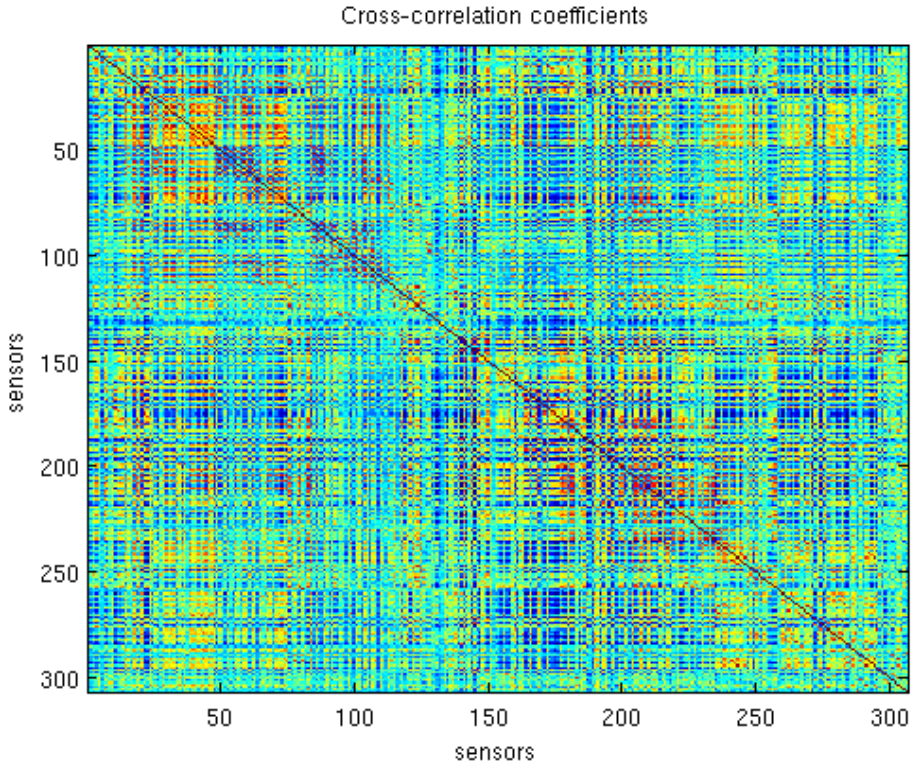

Figure 10: Max. Cross-correlation among features

this project will focus on predicting the particular word in a sentence by implementing the following strategies:

- Using different classifier that captures the the correlations accurately (e.g. k-NN)
- Implementing the classifier on GPU devices for higher performance and for tackling *Out of memory* errors.
- Using other functional connectivity methods such as Mutual Information and Cross-Coherence.

Acknowledgments

We thank N. Rafidi and D. Howarth for providing us with the dataset and their invaluable suggestions.

References

[1] Turk-Browne, Nicholas B. *Functional interactions as big data in the human brain* Science 342.6158 (2013): 580-584

[2] Gustavo Sudre, Dean Pomerleau, Mark Palatucci, Leila Wehbe, Alona Fyshe, Riitta Salmelin, & Tom Mitchell *Tracking neural coding of perceptual and semantic features of concrete nouns* NeuroImage, 62(1):451-463 (2012)

[3] Tom Mitchell *Predicting Human Brain Activity Associated with the Meaning of Nouns* AAAS Science 320, 1191 (2008)

[4] Leila Wehbe, Brian Murphy, Partha Talukdar, Alona Fyshe, Aaditya Ramdas, & Tom Mitchell *Simultaneously Uncovering the Patterns of Brain Regions Involved in Different Story Reading Subprocesses Plo*S one 9.11 (2014): e112575

[5] Rustandi, Indrayana, Marcel Adam Just, and Tom M. Mitchell. *Integrating multiple-study multiple-subject fMRI datasets using canonical correlation analysis* Proceedings of the MICCAI 2009 Workshop: Statistical modeling and detection issues in intra-and inter-subject functional MRI data analysis. Vol. 1. No. 5.4. 2009

[6] Wikipedia contributors. *"Cross-correlation* Wikipedia, The Free Encyclopedia. 28 Apr. 2015. Web. 5 May. 2015.