A Subjectivity Classification Framework for Sports Articles using Improved Cortical Algorithms

Article *in* Neural Computing and Applications · May 2018

DOI: 10.1007/s00521-018-3549-3

Yara Rizk
American University of Beirut
20 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Nadine Hajj
American University of Beirut
10 PUBLICATIONS **25** CITATIONS
SEE PROFILE

Mariette Awad
American University of Beirut
115 PUBLICATIONS **491** CITATIONS
SEE PROFILE

A Subjectivity Classification Framework for Sports Articles using Improved Cortical Algorithms

Nadine Hajj, Yara Rizk, Mariette Awad, PhD

American University of Beirut, Beirut, Lebanon

Abstract

The enormous number of articles published daily on the internet, by a diverse array of authors, often offers misleading or unwanted information, rendering activities such as sports betting riskier. As a result, extracting meaningful and reliable information from these sources becomes a time consuming and near impossible task. In this context, labeling articles as objective or subjective is not a simple natural language processing task because subjectivity can take several forms. With the rise of online sports betting due to the revolution in internet and mobile technology, an automated system capable of sifting through all this data and finding relevant sources in a reasonable amount of time, presents itself as a desirable and marketable product. In this work, we present a framework for the classification of sports articles composed of three stages: the first stage extracts articles from webpages using text extraction libraries, parses the text, and then tags words using Stanford's parts of speech tagger; the second stage extracts unique syntactic and semantic features, and reduces them using our modified cortical algorithm (CA) – hereafter CA* – while the third stage classifies these texts as objective or subjective. Our framework was tested on a database containing 1000 articles, manually labeled using Amazon's crowdsourcing tool, Mechanical Turk; and results using CA, CA*, support vector machines (SVM) and one of its soft computing variants (LMSVM) as classifiers were reported. A testing accuracy of 85.6% was achieved on a 4-fold cross validation with a 40% reduction in features using CA* that was trained using an entropy weight update rule and a cross entropy cost function.

Keywords: Natural language processing; Subjectivity analysis; Feature reduction; Cortical algorithm; Support vector machines

1. Introduction

With the increasing interest in betting on sports, and with the rise of internet and mobile technology, revenues from this form of gambling is projected to remain on the rise. For instance, betting on the 2013 Super Bowl, in Nevada alone, saw $98.9 million in wagers, a 5% increase from last year's Super Bowl[1]. Bets placed using mobile devices increased by 114%, which resulted in 98% increase in revenue[2]. Mobile apps have also been developed to help people place and track bets[3], and allow peer to peer wagering[4].

To increase their chances of placing winning bets, users need to rely on accurate and dependable expert sources, which due to the enormous number of online articles published daily by a diverse array of authors, become hard if not impossible to identify. Hence, an automated solution with the ability to distinguish between objective and subjective articles presents itself as a desirable and marketable product.

However, this task is challenging due to the different aspects of subjectivity in a text, involving not only machine intelligence but also a significant level of information analysis and extraction. Subjectivity can be expressed in a multitude of ways: directly using statements such as "I believe" and "I think" or inferred. Another challenge in subjectivity analysis lies in the fact that different sentence structures and different words and phrases can convey varying intensities depending on the context which includes neighboring words and punctuation. Furthermore, articles need not be entirely subjective or objective. Many authors incorporate few objective statements to support their personal beliefs. Also, some subjective statements become widely accepted as facts and make it more difficult for a software tool to correctly classify articles. As a result, millions of features have been extracted in the literature to find the best model for this prediction. From syntactic to semantic features including parts of speech, n-grams, word level sentiment and phrase level sentiment scores, as more features are included in the model, the prediction time and memory requirements increase as well. Reducing the number of these features to create efficient predictors for online applications without greatly reducing their accuracy adds to the already challenging text based sentiment analysis problem.

In this paper, a framework capable of classifying sports-related text into objective and subjective categories based on a unique combination of syntactic and semantic features, as well as an adaptive feature reduction scheme using modified CA, is proposed. While CA has been proposed in the literature as a machine learning (ML) technique showing promise for classification problems (Hashmi et al. 2009, 2010), we modify it and apply it to feature reduction. CA hierarchically extracts features from input data identifying components in an increasing complexity scheme, where lower layers learn simple aspects of the data and upper layers provide a more complex level of abstraction. Consequently, the input layer scheme becomes a strong indicator of the significance of the features fed into the network and features with a significance factor below a predefined threshold can be eliminated.

Feature selection was based on inputs from some linguistic specialist and feature reduction was performed adaptively using a CA trained using a weighted entropy cost function and the well-formed cross entropy (CE) cost function, hereafter CA*. Several classifiers were trained on the resulting feature set to perform the classification task, including, CA, CA*, support vector machines (SVM) and one of our earlier soft computing SVM variants, LMSVM (Rizk et al. 2013). LMSVM attempts to reduce the number of support vectors (SVs) used to represent SVM's separating hyper-plane for energy aware computing in limited resource platforms. LMSVM selects fewer SVs because it retains, after clustering, points near the boundary to train SVM and produces an approximation of SVM's more complex separating hyper-plane. Tested on a corpus of 1000 texts gathered from various websites and labeled using Amazon's Mechanical Turk crowdsourcing tool, our framework obtained results that motivate follow on research.

In this work, we have made the following contributions. The sports articles' dataset for subjectivity analysis used in (Rizk and Awad, 2012) was extended to include a larger number of diverse articles. Novel combination of semantic and non-semantic features was employed then reduced by CA*, and multiple classifiers for the task of subjectivity analysis for sports articles were compared.

The rest of this paper is structured such that a literature review on related work in sentiment text analysis and feature reduction is presented in Section 2. Section 3 includes a discussion on the proposed framework whereas Section 4 presents the results of the performed experiments. Finally, conclusions based on our findings are drawn in Section 5.

2. Literature review

Many researchers have attempted to identify text subjectivity or sentiment analysis using NLP techniques. First, some of the proposed algorithms to classify documents as objective or subjective are discussed in subsection 2.1. Then, some feature reduction algorithms, proposed in literature, are presented in subsection 2.2.

2.1. NLP based subjectivity analysis

SENTIWORDNET (Esuli and Sebastiani 2006) is an algorithm that assigns a positive subjectivity score, a negative subjectivity score and an objectivity score to a given word. This is accomplished by a set of SVM and Rocchio classifiers that classify a word into one of three classes: objective, positive or negative. The overall score of each word is based on a combination of the scores assigned to it by the various classifiers. Words were represented by a feature vector generated from the gloss of a word. The classifiers were trained on different word sets, in a semi-supervised fashion, i.e. some of the words were manually labeled. Authors in (Yu and Hatzivassiloglou 2003) also attempted to assign positive, negative and neutral tags to words. Given seed words with pre-assigned polarity scores, new words are tagged with labels similar to the seed words that have a high co-occurrence measure.

Considering sentiment analysis at a sentence or phrase level, Yu and Hatzivassiloglou (2003) developed three algorithms to label sentences from Wall Street Journal articles. The first method measured sentence similarity using features, such as common synsets, by identifying the topic of the article and comparing the sentences in the given article to other articles in the same topic category. The second method used a Naïve Bayes classifier to classify sentences based on parts of speech (POS), n-grams (n = 1, 2 and 3), word-based semantic orientation count, sequences of semantic orientations and other features. Finally, the third method, presented in (Yu and Hatzivassiloglou 2003), combined several Naïve Bayes classifiers trained on distinct feature sets which are subsets of the features used in the second method.

Wilson et al. (2005) used contextual information to identify contextual polarity of sentences. First, a subjectivity lexicon was created by the authors, containing more than 8000 words. This lexicon was used to generate features for the neutral-polar classification task and then the positive-negative classification task. Some features used in the former task, as reported in Wilson et al. 2005, include word POS, prior word polarity, preceded by adjective, adverb or intensifier, and adjective/adverb count in sentences…

Wiebe and Riloff (2011) attempted to use information extraction methods to improve word and expression subjectivity classification by parsing the texts into clauses and identifying their syntactic features. Then, bootstrapping algorithms were used to learn subjective nouns. Subjective expressions were learnt using AutoSlog-TS and lexico-syntactic patterns previously extracted. Finally, a Naïve Bayes classifier was used to classify sentences from unlabeled texts.

Unlike the presented work above, we do not attempt to classify words or phrases but entire documents instead. In some cases, sentiment or opinion analysis of individual words or sentences is used to aid in the document-level classification.

Publications that addressed the document-level classification task include (Das and Bandyopadhyay 2010) who used genetic algorithm (GA) on a set of syntactic, semantic and discourse features extracted from news articles to select the feature set and classify them into objective or subjective articles. The features included the frequency of parts of speech, word based SENTIWORDNET scores, word stems, document title, position of subjectivity indicators, term distribution among others. Their feature set and GA classifier achieved 90.22% on the English news corpus and 93% on the Movie Review corpus.

In our earlier work (Rizk and Awad 2012), sports articles were classified as objective or subjective using only a set of frequentist syntactic features, which included punctuation, numerical values and dates, and POS tags produced by Stanford POS Tagger (Toutanova et al. 2003). A GA was trained on a homemade database of 300 sports articles and achieved an accuracy of 96.2% on a 3-fold cross fold validation scheme.

Heerschop et al. (2011a) computed a document score based on lexicons. Texts were parsed to sentences then words followed by POS tagging and sense disambiguation. A sentiment score was assigned to a word using algorithms such as SENTIWORDNET and PageRankSWN. Finally, SVM and Rocchio classifiers were used to generate a final sentiment class.

Pang et al. (2002) classified movie reviews into objective and subjective classes, using Naïve Bayes, SVM and maximum entropy classifiers. Extracted features included POS of words, unigrams, bigrams, adjectives…

Although (Pang et al. 2002) only extracted unigrams and bigrams, other researchers investigated extracting n-grams with n $>$ 2. Since N-grams generates many potentially redundant and noisy features, Abbasi et al. (2011) presented a feature selection algorithm to reduce the number of these features, enabling the use of n-grams to produce better sentiment classification accuracy. This algorithm, called Feature Relation Network (FRN), contained 22 n-gram feature groups connected by parallel relations. The algorithm computed weights for these features and eliminated weak or redundant features. The remaining features were given to a linear kernel SVM classifier.

Wiebe et al. (2002) used a k-nearest neighbor to classify articles. The distance was calculated by simply subtracting the potential subjective element (PSE) count of the documents, which was normalized by the word count. PSEs include verbs, adjective, adverbs, unique words, modals… that appear in subjective texts of the training corpus, as stated in (Wiebe et al. 2002).

Yu and Hatzivassiloglou (2003) used a Naïve Bayes classifier given un-stemmed unigrams from the Wall Street Journal corpus. Devitt and Ahmad (2007) tackled the problem of news article sentiment classification by combining syntactic and semantic features including POS, punctuation and SENTIWORDNET-based lexical affect metric.

Godbole et al. (2007) broke down texts into distinct entities using Wordnet and computed a polarity (positive and negative) score for each entity. They also measured the positive or negative portrayal of individuals and issues in news articles vs. blogs.

Finally, some work investigated the impact of specific features; Heerschop et al. (2011b) investigated the impact of negation, whereas Benamara et al. (2007) looked at the effect of adverbs and adjective-adverb combinations.

While previous sentiment detection models are not optimized for real-time performance, many models require days and even weeks to run (Wang et al. 2012), recent models have been developed to achieve efficient computation. Wang et al. (2012) developed a Twitter sentiment analysis system that analyses the positivity or negativity of election candidate related tweets. Their system can process 72 tweets per minute and correctly classified 59% of the tweets into positive, negative, neutral and unsure. Guerra et al. (2011) adopted a transfer learning approach to develop a real-time sentiment analysis system applied to tweets and produced prediction accuracies between 80% and 90%.

2.2. Feature reduction

Since the number of features in NLP can be significantly large, few feature reduction algorithms have been applied to NLP corpuses. Below, we mention only some of the most relevant works.

Berger et al. (1996) proposed a maximum entropy approach to NLP applications with an entropy-based scheme for selecting features. Starting with an empty set, and at each iteration step, training adds one feature to the available set and the added feature is retained if the new model achieves a higher performance, otherwise the feature is discarded.

Yang et al. (1997) conducted a comparative study of five feature selection techniques in the context of text categorization: document frequency (DF), information gain (IG), mutual information (MI), chi-squared test (CHI) and term strength (TS). IG discarded features that did not add information to the ML model while MI discarded the ones that fell below a mutual information criterion that describes the information added by a certain term to a certain category across all the documents. CHI computed a normalized measure of the independence between terms and categories and eliminated the features that are independent with respect to a certain category and TS measured the likelihood of the presence of a certain term in a particular category and discarded features below a pre-defined threshold. DF was found to be the most computationally efficient with acceptable performance, while the other techniques performed similarly better at the expense of an increased complexity.

Liu et al. (2003) proposed an iterative feature selection method based on expectation maximization (EM). Their algorithm finds the optimal subset that maximizes a likelihood function given the probability of all documents for the feature set. The E-step calculates the expected subset of features relevance given the results of a K-means clustering algorithm, while the M-step calculates the maximum likelihood estimation of the clustering result in the new feature space.

Kim et al. (2005) suggested three dimensionality reduction method for text classification: a centroid algorithm that minimizes the distance between the centroids of the actual data clusters and the centroids of the clusters in the reduced space, an orthogonal centroid algorithm that computes the reduced dimension space based on the QR decomposition of the original data, and a generalized linear discriminant analysis based on singular value decomposition that conserves the clusters shape in the reduced dimensionality space.

Shafiei et al. (2007) investigated three feature reduction techniques for text clustering: independent component analysis (ICA), latent semantic indexing (LSI) and document frequency selection (DF). ICA looked for a reduced space where the original set of features was the result of linear or nonlinear mixtures of the reduced feature vectors under the assumption of a Gaussian distribution. LSI was based on singular value decomposition (SVD), where a portion of the singular values is set to zero to obtain a partial SVD decomposition of the data corresponding to the reduced feature space while the DF selection method relied on the simple intuition inferred by keywords and features occurrences in documents.

Chua (2009) researched the Latent Dirichlet Allocation probabilistic model for feature reduction for unsupervised document clustering. This method found the optimal Dirichlet distribution that maximized the joint probability of observing a corpus in a document using Gibbs sampling and the EM algorithm to maximize the likelihood of the occurrence of a term in a category of documents.

Mao et al. (2010) used domain knowledge to reduce the feature vectors for text applications. The unsupervised proposed method computes a distance measure between documents using three ways: a manual technique, a corpus statistic technique and a linguistic expert based technique. The geometry obtained by computing the distances between documents is then used to derive a reduced feature space.

Feature reduction has been achieved using linear and non-linear algorithms; the former includes principle component analysis (PCA), linear discriminant analysis (LDA) and SVD, whereas the latter includes GA, SVM and neural networks.

Bian et al. (2011) reduced the number of features in input space or kernel space by proposing a max-min distance analysis (MMDA) criterion, which maximizes the minimum interclass distances in the reduced space. Tang et al. (2005) incorporated a weight interclass dependence factor into the intra-class scatter matrix of LDA, improving its performance. Chen et al. (2011) selected a representative subset of features by traversing an ordered tree, composed of nodes representing all the possible reduced sets, and pruning nodes to obtain an optimal or approximate solution. Han et al. (2012) attempted to improve feature selection stability by giving weights to training points calculated based on some feature relevance metric.

Raymer et al. (2000) reduced the number of features by training a GA in a supervised fashion, obtaining a weight vector, using the weights as a ranking of the features and iteratively eliminating the least important features until maximum prediction accuracy is attained. Encoding the feature indices into GA's chromosome and using the prediction accuracy of a classifier as a fitness value allows GA to perform feature reduction (Atyabi et al. 2012). Particle swarm optimization can be used to perform feature reduction by searching for a subset that best represents the data in a given sub-epoch (Atyabi et al. 2012). Random search retains the feature subset with highest prediction accuracy and randomly generates a new population, in each iteration, until termination (Atyabi et al. 2012).

Perantonis and Virvilis (1999) trained a neural network on the entire feature set multiple times with various starting points. The reduced feature set consists of the eigenvectors of the saliency matrix computed based on the trained neural network. Hinton and Salakhutdinov (2006) trained a multilayer neural network with a small central layer using gradient descent as a learning algorithm to perform feature reduction. Deepthi et al. (2007) adopted multi-layer mirroring neural networks to reduce the dimensionality of a database after initializing the weights and bias terms to either weak or random values.

Bi et al. (2003) ranked the features using a series of cascaded sparse and linear SVMs based on some computed weighted variables. Wang et al. (2010) performed feature reduction in an unsupervised manner by computing kernel-based measures of independence, which included the radial basis network, random space feature and a non-parametric model. Finally, Formisano et al. (2008) applied a feature extraction algorithm based on a representation of magnetic resonance images of the brain of individuals speaking three different Dutch phonemes and used an SVM to classify the Dutch phonemes based on these extracted features.

3. Proposed framework

3.1. Overview

We followed the generic flow of Fig. 1 in our framework. After the text is extracted from webpages, it is preprocessed to remove any undesirable content that might hinder the correct execution of the remaining blocks. The extracted features can be divided into the blocks shown in Fig. 2. The text is tagged by Stanford POS tagger (Toutanova et al. 2003) and then padded with our own additional functions that distinguish the various pronoun (first, second and third) classes and extract additional information about verb tenses (e.g. imperative versus base form verbs). The frequencies of various POS that are of interest in the subjective/objective classification problem form the syntactic features. Semantic information from the text is extracted using SENTIWORDNET (Esuli and Sebastiani 2006). The opinion contents of the first and last sentences in the text are also added as features to classify the overall text. A classifier labels the given article as objective or subjective based on these extracted features. Several classifiers were tested including a CA, SVM and some of their variants.

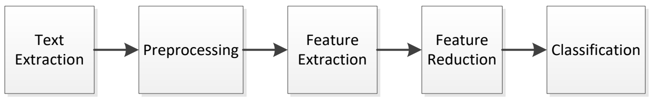

Fig. 1. Workflow of automated subjectivity analyzer

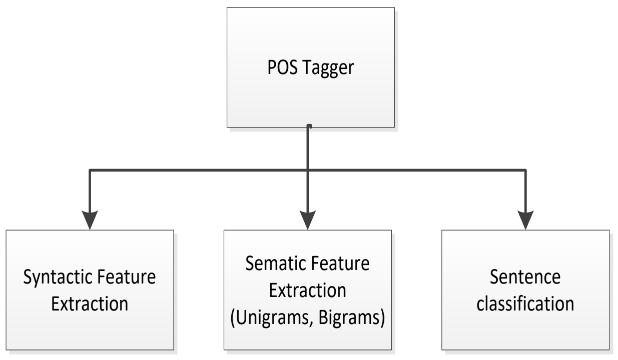
POS Tagger
Syntactic Feature Extraction
Sematic Feature Extraction (Unigrams, Bigrams)
Sentence classification

Fig. 2. Feature extraction workflow

3.2. Feature generation

Fifty-two features, listed in Table 1, were extracted from the text and used as input to the classifier, based on few discussions with English linguists at the American University of Beirut.

The features used in this affective computing framework can be divided into two categories. The first category includes all syntactic features that were extracted from the text. The model computes the frequency of various POS such as pronouns and verb tenses, and punctuation such as quotations and question marks. Features 1 through 36 are a simple count of tags that are extracted by the Stanford POS Tagger (Toutanova et al. 2003). Features 37 through 48 are a count of a combination of tags or a subdivision of the tags. A subset of these features was intuitively divided into two classes, those that are most likely indicative of objectivity and those indicative of subjectivity as shown in Table 2. The detailed justification for these feature choices and divisions can be found in our earlier work (Rizk and Awad 2012).

The second category refers to semantic content and includes features 49 through 52 in Table 1. The word-based subjectivity and objectivity semantic scores give a measure of the number of objective and subjective words in the article. These features are computed in the following manner. A sentiment score is assigned to each word in the text using SENTIWORDNET (Esuli and Sebastiani 2006); each word receives a positive, negative and objective score which is between 0 and 1. The positive, negative and objective scores are summed over all the words in the text and used to normalize the individual scores of each word. The normalized scores are compared to a threshold. If a word has the positive or negative score greater than this threshold, the subjective words counter is incremented. On the other hand, if the objective score is greater than this threshold, the objective words counter is incremented.

The classes of the first and last sentences in an article were included in the feature vector as well. Since the first sentence in a text usually states the writer's purpose, therefore, if the author's goal is to simply narrate, i.e. write an objective article, then the first sentence will most likely be an objective sentence. The last sentence usually contains the author's concluding remarks about ideas discussed in the article. Authors can also reiterate the position they were defending throughout the text. Therefore, a subjective text is more likely to conclude with a subjective sentence. The sentence is classified by simply comparing the count of sentiment oriented words versus objective or neutral words, excluding stop words (prepositions, determiners…).

Table 1. List of generated features

1	Frequency of coordinating conjunctions
2	Frequency of numerals and cardinals
3	Frequency of determiners
4	Frequency of existential there
5	Frequency of foreign words
6	Frequency of subordinating preposition or conjunction
7	Frequency of ordinal adjectives or numerals
8	Frequency of comparative adjectives
9	Frequency of superlative adjectives
10	Frequency of list item markers
11	Frequency of modal auxiliaries
12	Frequency of singular common nouns
13	Frequency of singular proper nouns
14	Frequency of plural proper nouns
15	Frequency of plural common nouns
16	Frequency of pre-determiners
17	Frequency of genitive markers
18	Frequency of personal pronouns
19	Frequency of possessive pronouns
20	Frequency of adverbs
21	Frequency of comparative adverbs
22	Frequency of superlative adverbs
23	Frequency of particles
24	Frequency of symbols
25	Frequency of "to" as preposition or infinitive marker
26	Frequency of interjections
27	Frequency of base form verbs
28	Frequency of past tense verbs
29	Frequency of present participle or gerund verbs
30	Frequency of past participle verbs
31	Frequency of present tense verbs with plural 3rd person subjects
32	Frequency of present tense verbs with singular 3rd person subjects
33	Frequency of WH-determiners
34	Frequency of WH-pronouns
35	Frequency of possessive WH-pronouns
36	Frequency of WH-adverbs
37	Frequency of quotation pairs in the entire article
38	Frequency of questions marks in the entire article
39	Frequency of exclamation marks in the entire article
40	Frequency of first person pronouns (personal and possessive)
41	Frequency of second person pronouns (personal and possessive)
42	Frequency of third person pronouns (personal and possessive)
43	Frequency of comparative and superlative adjectives and adverbs
44	Frequency of past tense verbs with 1st and 2nd person pronouns
45	Frequency of imperative verbs
46	Frequency of infinitive verbs (base form verbs preceded by “to”)
47	Frequency of present tense verbs with 3rd person pronouns
48	Frequency of present tense verbs with 1st and 2nd person pronouns
49	Frequency of words with an objective SENTIWORDNET score
50	Frequency of words with a subjective SENTIWORDNET score
51	First sentence class
52	Last sentence class

Table 2. Objective and subjective features

Features			Objective	Subjective
Syntactic	Punctuation	Quotations	X	

		Question marks			X
		Exclamation marks			X
	Pronouns	First & Second person			X
		Third person	X		
		Past tense	X		
	Verb Tenses	Imperative tense			X
		Present tense		X	
	Adjectives & Adverbs	Comparative & Superlative			X
Semantic	Objective words		X		
	Subjective words				X
Sentences	First sentence			X	
	Last sentence			X	

3.3. Feature reduction

A robust ML model capable of effectively classifying sports articles requires highly discriminative features. Because the large variability in subjectivity indicators and authors' expertise could render the feature extraction stage noise, we propose to use CA* for feature reduction to find the optimal subset of features for this affective computing task.

3.3.1. Cortical algorithm description

Modeled after the human brain, CA are a biologically inspired ML approach modeled by MountCastle after the human visual cortex that store sequences of patterns, recall patterns auto-associatively, store patterns in an invariant form hierarchically. Originally described by Edelman and Mountcastle and further developed by (Hashmi and Lipasti 2009-2010), CA are six-layered structures, each layer having a different thickness and containing a very large number of neurons strongly connected via feed-forward and feedback connections (Edelman and Mountcastle 1982). The mini-column, a vertically shaped structure, is the basic unit in a cortical network; its counterpart is the neuron in a classical artificial neural network. It is a group of neurons that share the same receptive field, i.e. associated to the same sensory input region. In what follows, mini-columns will be referred to as columns. An association of columns is referred to as hyper-column or layer. The connections in the network occur in two directions: vertical connections between columns of consecutive layers and horizontal connections between columns within the same layer. Horizontal connections are also known as lateral inhibiting connections; these connections don't represent data propagated between neurons but serve as a communication means between these columns. An illustration of the topology of a CA network is shown in Fig. 3.

Training a cortical network occurs in 3 phases: random initialization, unsupervised feed-forward learning and supervised feedback training.

1. Random Initialization: Initially, all the synaptic weighs are initialized to random values that are very close to 0; hence, no preference for any particular pattern is shown.
2. Unsupervised Feed-forward: During the feed-forward learning stage, columns are trained to identify features not detected by others, by random firing and repeated exposure. This objective is achieved by enforcing the random firing of a particular column for a specific input pattern. When a pattern is presented to the network, the firing columns strengthen their weights using the strengthening weight update rule in (1); (1) adds an exponential value to the column's weights. Simultaneously, these firing columns inhibit neighboring columns within the same layer from firing by weakening their weights using the inhibiting update rule in (2); (2) subtracts (3) from each of the column's weights. Eq. (3) computes $\Omega\left(W_i^{r,t}\right)$, a value

proportional to the summation of a column’s weights multiplied by the column’s input value is subtracted from each of the column’s weights.
$W^{r,t}_{i,j,k}$ denotes the weight of the connection between the jth neuron of the ith column of layer r and the kth column of the previous layer $(r-1)$ during the training epoch t. $Z^{r,t}_{k}$ denotes the output of the kth column of the rth layer during the training epoch t. ρ is a tuning parameter and ε is the firing threshold; both parameters, chosen empirically, are constant for all epochs and columns.

3. Supervised Feedback: The supervised feedback has a goal to correct misclassifications of the same pattern. Another variation of the same pattern that is quite different from the previous one, may lead to a misclassification when the top layer columns that are supposed to fire for that pattern do not fire. To reach a desired firing scheme over multiple exposures also known as “stable activation”, the error occurring at the top layer generates a feedback signal forcing the column firing for the original pattern to fire and inhibiting the column that is firing for the new variation. The feedback signal is propagated back to the previous layer once the top-level columns start to give a stable activation for all variations of the same pattern. The feedback signal is sent to preceding layers only once the error in the layer concerned converges to a value below a certain pre-defined tolerance threshold. Each layer is trained until a convergence criteria expressed as an error term in function of the actual output and a desired output (scheme of firing) is reached. The excitatory and inhibiting signals follow the same update rules as the feed-forward learning.

$$\begin{cases} W^{r,t+1}_{i,j,k} = \alpha W^{r,t}_{i,j,k} + \theta\left(W^{r,t}_{i,j,k}\right) \\ \theta\left(W^{r,t}_{i,j,k}\right) = \beta \Big/ \left(1 + \exp \frac{W^{r,t}_{i,j,k} - T}{\Omega\left(W^{r,t}_{i}\right)}\right) + \delta \\ \alpha = Z^{r-1,t}_{k};\ \beta = Z^{r-1,t}_{k}.\rho;\ \delta = Z^{r-1,t}_{k}.C^{r,t}_{i,j,k} \\ C^{r,t}_{i,j,k} = \begin{cases} 1 \ if\ W^{r,t}_{i,j,k} > \epsilon \\ 0\ otherwise \end{cases} \end{cases} \tag{1}$$

$$W^{r,t+1}_{i,j,k} = Z^{r-1,t}_{k}\left(W^{r,t}_{i,j,k} - \Omega\left(W^{r,t}_{i}\right)\right) \tag{2}$$

$$\Omega\left(W^{r,t}_{i}\right) = \sum_{j=1}^{M}\sum_{k=1}^{L_{r-1}} C^{r,t}_{i,j,k} W^{r,t}_{i,j,k} \tag{3}$$

In our earlier work (Hajj and Awad 2013), we had proposed CA*, a weighted entropy weight update rule to train CA along with the use of the well-formed cross entropy cost function, which lead to an improved performance of the algorithm at the expense of a minor degradation in the training time. The inhibiting and strengthening weighted entropy weight update rules are given by (4) and (5), respectively, where $J^{r,t}$ is the well-formed CE cost function associated with the rth layer at the training epoch t, and Δ stands for the gradient operator.

$$W^{r,t+1}_{i,j,k} = Z^{r-1,t}_{k}\left(W^{r,t}_{i,j,k} - \frac{\Delta J^{r,t}}{\Delta Z^{r,t}_{i}}\Omega\left(W^{r,t}_{i}\right)\right) \tag{4}$$

$$W^{r,t+1}_{i,j,k} = \alpha W^{r,t}_{i,j,k} + \frac{\Delta J^{r,t}}{\Delta Z^{r,t}_{i}}\left(\beta \frac{1}{1+\exp\left[\frac{W^{r,t}_{i,j,k} - T}{\Omega\left(W^{r,t}_{i}\right)}\right]} + \delta\right) \tag{5}$$

Unlike the mean squared error which is CA’ original cost function, CE interprets the outputs as probabilities maximizing the mutual information between the actual output and the desired output. In addition, the CE cost function was proven to achieve lower error rates in imbalanced datasets and reduced convergence rates to local minima for large networks (Silva et al. 2005, 2008), making it highly desirable for our affective computing scenario.

In this work, we adaptively employ the weighted entropy update rule and the well-formed cross entropy cost function, to reduce the feature set obtained during the feature generation stage – which to the best of our knowledge has not been investigated yet. The following subsection details the proposed technique.

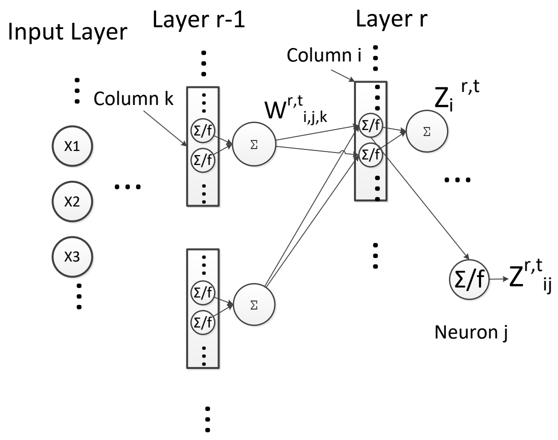

Fig. 3 Cortical Network Architecture

CA exhibits several plausible properties that allow it to outperform other classification techniques.

- A "smart storage" ability, evident in the firing rate: despite the large number of hyperparameters the CA possesses (synaptic connections' strength), the firing rate which quantizes the percentage of connections "alive" after training shows that instances are stored in a very small subsection of the network, hence, allowing for more patterns to be stored with overlap in storage mapping that in similarities between stored instances.
- An invariant representation storage: while SVM only looks for a separating hyperplane that divides the training data into different classes, CA attempts to create robust invariant representations of data points belonging to the same class, hence, emphasizing differences between classes while maintaining robustness with respect to within-class variations.
- Input layer connections extract meaningful attributes while weakening less discriminative ones: as shown in feature reduction results, the input layer's weights reflect the significance of the features in creating unique representations. While KL and PCA only perform statistical analysis to rank and eliminate features, CA relies on the learnt input weights at convergence to discard less significant attributes.
- Hierarchical learning: The two-stage learning algorithm that CA uses trains different columns within the network to extract and respond to particular aspects of the input data, this not only allows the network to create a hierarchical representation of the instance using increasingly complex characteristics but also to identify important patterns in the data.

3.3.2. Cortical algorithm for feature selection

As mentioned earlier, one of the advantages of CA is their ability to learn aspects of the data in a hierarchical manner. Layers in a cortical network identify components of the input data in an increasing complexity scheme: lower layers learn simpler characteristics and upper layers build a more complex level of abstraction.

During feed-forward unsupervised learning, cortical columns that fire for specific data patterns are enforced as shown in the strengthening rule. Through repeated exposure, particular columns are trained to fire for particular aspects of the data. During the supervised feedback learning, invariant representations are created, i.e. the network is trained to recognize different versions of the same pattern as one. The resultant of this training procedure is a network that can detect discriminative aspects of the data that distinguishes a class from another while maintaining certain robustness against disparities within the same class. One important asset of this learning scheme is its ability to extract meaningful information from the input data and detect less important or noisy data. As a result, less discriminative features are associated with weak weights while other significant features are accounted for in the form of stronger weights. Thus, the weights of the input layer directly connected to the feature vector are an indication of the significance of the corresponding features. This property gives a cortical network the ability to discriminate and identify meaningful features.

Our method employs the input layer firing scheme to rank the feature vector in terms of significance. In other terms, after training the cortical network using the given set of inputs, the features are ranked according to their corresponding average input weight. The most significant features are associated with the highest weight values while less important features are weighted with weak values. We discard a percentage of the lowest features i.e. those linked to the weakest input weight values. The process is repeated sequentially and results using multiple feature reduction ratios are observed. The subset corresponding to the best performance is retained. In the experimental result section, we detail the performance of 3 classifiers on increasing reduction ratios to illustrate the effect of feature reduction. The following illustrates our proposed workflow:

1) Train the cortical network using all features provided.
2) For all input nodes, calculate the average weight associated with each feature f (fth input column), $\mathrm{W_f} = \frac{1}{\mathrm{LK}}\sum_{\mathrm{i}=1}^{\mathrm{L}}\sum_{\mathrm{j}=1}^{\mathrm{K}} \mathrm{W}_{\mathrm{i,j,f}}^{1}$, L representing the number of columns in the 1st layer, K representing the number of neurons per column, and 1 referring to the 1st layer.
3) Rank features in a decreasing order according to their associated weights.
4) Discard a percentage of the lowest ranked features (10% in our experiments).
5) Repeat steps (1) till (4) using the new subset obtained.
6) Retain the subset corresponding to the best performance of the algorithm.

3.4. Classifiers

Several classification algorithms were compared: a generic SVM with a Gaussian kernel using the libSVM library (Chang and Lin 2011), LMSVM (Rizk et al. 2013), CA* and a regular CA trained using the mean squared error (MSE) cost function and Lipasti's exponential weight rule.

4. Experimental results

4.1. Corpus description

A corpus, composed of 1,000 English sports articles, was gathered from over 50 different websites including NBA.com, Fox Sports, and Eurosport UK. Most of these articles were written by professional journalists but a few were written by avid sports fans on sports blogs. The articles ranged in length from 47 to 4283 words, with an average length of 697 words. This corpus was labeled using Amazon's Mechanical Turk, a crowd sourcing tool. This labeling was used as ground truth to compare the performance of our algorithm. 658 articles were labeled objective and 342 were labeled subjective. Unlike the Wall Street Journal database which is labeled based on its metadata information and originated from a single source, written by professional journalists, our database is unbalanced and composed of articles from various sources and authors. Therefore, the linguistic structures of the articles vary significantly, making the classification task harder.

4.2. Experimental setup

All experiments are based on a 4-fold cross fold validation scheme that exposes all feature reduction and classification algorithms to the database division. First, the database passed through a preprocessing phase that eliminated 3 features because their values are redundant for all classes hence possessed no discriminative ability among classes. The remaining 49 features were passed through the feature reduction phase where one of the following algorithms incrementally eliminated 10% of the least significant features, features before training a classifier on the data. CA's feature reduction performance was compared to two widely used algorithms in the feature reduction literature, principle component analysis (PCA) and Kullback-Leibler distance ranking criteria (KL). The former represents observations in the principal component space features with the lowest eigenvalues whereas the latter computes the relative entropy of features and ranks them to in terms of discriminative ability and eliminates those features with low discriminatory abilities.

Four classifiers were used to assess the performance of the feature reduction schemes, CA*, CA, SVM and LMSVM (Rizk et al. 2013). SVM and LMSVM were trained after applying the RBF kernel to the input data. A grid search for each feature set was performed to obtain the model parameters of SVM and LMSVM which include box constraint and RBF's sigma for both classifiers and the pullback parameter that determines the data reduction in LMSVM. The CA network was composed of 6 layers consisting of 20-node columns starting with 2000 columns in the first layer and reducing this number by half between consecutive layers. This network topology was fixed for all experiments.

4.3. Feature reduction and classification results

To test the performance of CA for feature reduction, we compare the classification results obtained against the ones generated by PCA and KL. On the other hand, to evaluate CA as a classifier, SVM and LMSVM were used to benchmark CA and CA*'s performance. The classification results, using four classifiers, for various reduction rates from the three feature reduction algorithms are summarized in Table 3. Also shown in this table is the relative improvement of CA* over each of the classifiers for different reduction rates. The best recognition rate for each experimental setting is high-lighted, and the best, worse and average performances are shown in italic.

The following observations can be drawn from our results:

1) Averaging over all classifiers to compare the feature reduction algorithms, CA (83.25%) outperformed KL (82.5%) and PCA (80.2%). This shows the ability of CA to robustly discriminate features.
2) The best performance of our proposed feature reduction scheme was achieved when using a CA* classifier on 30 features (85.7%). This feature subset led to a lower recognition rate for CA (84.5%), SVM (82%) and LMSVM (82.1%). CA* achieved an accuracy of 84.7% when KL selected only 50% of the features, whereas CA, SVM and LMSVM achieved accuracies of 84%, 82.5% and 82.2%, respectively. CA* achieved an accuracy of 83.7% when PCA eliminated 40% of the. Combining PCA with SVM and LMSVM achieved accuracies of 82.1% and 82.3%, respectively. Even though the highest accuracies for KL and PCA were obtained for a higher reduction rate, these accuracies were still lower than CA's accuracies.
3) Without feature reduction, the CA* classifier (84.8%) outperformed CA (83.1%), SVM (80.3%) and LMSVM (80.3%). However, this is lower than the best accuracy achieved on this database which reflects the importance of feature reduction in eliminating noisy features for all classifiers.
4) Training SVM on 5 attributes only, the features selected by CA resulted in an accuracy of 83%, whereas those selected by KL and PCA attained 78.7% and 75.5%, respectively further validating the quality of the selected features by CA even in extreme reduction scenarios.
5) When comparing classifiers, CA* outperformed other classifiers by achieving an average improvement of 1.09% over CA, 2.79% over SVM and 2.82% over LMSVM. More specifically, using CA for feature reduction, the relative average improvement of CA* over CA is 1.1% and 2.53% over SVM and LMSVM. For a reduction rate of 10%, CA outperformed SVM and LMSVM by 4.7% and 4.9%, respectively. A similar analysis using KL for feature reduction shows a maximal improvement for CA* of 1.8% vs CA, 4.2% versus SVM, and 4.7% versus LMSVM. Using PCA, the maximal improvement for CA* is of 2%, 11.6%, 11.4% over CA, SVM and LMSVM, respectively.

The effect of the features on the number of SVs for SVM and LMSVM is summarized in Table 4. Also shown in this table are the reduction in number of SVs and the relative improvement in performance of LMSVM vs SVM. We observed that feature reduction lead to a reduction in SVs for both algorithms. CA produced an average across various reduction rates of 480 and 463 SVs for SVM and LMSVM respectively, whereas KL lead to 504 and 475 SVs and PCA lead to 553 and 501 SVs. This result further validates CA's merit as a feature reduction algorithm since less SV indicate a more linearly separable representation of the data in the kernel space after eliminating noise features using the various feature reduction algorithms.

Focusing on features, the following observations can be made. The most significant features according to the results generated using CA are the frequency of second person pronouns (personal and possessive), the frequency of personal pronouns and the frequency of superlative adverbs as they constitute a strong indicator of the subjectivity of the article. The least significant features are frequency of plural proper nouns, the frequency of numerals and cardinals and the frequency of past tense verbs with 1st and 2nd person pronouns as they are widely used in both classes of articles. These results are consistent with our intuition that pronouns and superlative adverbs indicate subjectivity and that past tense verbs, proper nouns, numerals and cardinals do not have a great influence. However, past tense verbs with 1st and 2nd person pronouns could have been considered indicators of subjectivity, contrary to CA's conclusions. This discrepancy can be attributed to different writing styles where some authors tend to use first person pronouns to express personal views or narrating events.

Table 3: Classification results

Feature Reduction Algorithm	Feature Reduction (%)	Number of Features	Recognition Rate (%)				Relative Improvement of CA* (%)		
			CA*	CA	SVM	LMSVM	(CA* – CA)/CA*	(CA* – SVM)/CA*	(CA* – LMSVM)/CA*
CA	0	49	84.8	83.1	80.3	80.3	2	5.31	5.31
	10	45	85.7	83.7	81	80.8	2.33	5.48	5.72
	20	40	85.2	83.9	81.7	81.6	1.53	4.11	4.23
	30	35	85.5	84.1	81.7	81.6	1.64	4.44	4.56
	40	30	85.6	84.5	82	82.1	1.29	4.21	4.09
	50	25	85.1	84.2	83.1	83.2	1.06	2.35	2.23
	60	20	85.2	83.6	82.5	82.7	1.88	3.17	2.93
	70	15	83.6	83.0	83.7	83.7	0.72	-0.12	-0.12
	80	10	83.1	82.8	82.1	82.1	0.36	1.2	1.2
	90	5	82.6	82.5	83	83.0	0.12	-0.48	-0.48
	Minimum		*82.6*	*82.8*	*80.3*	*80.3*	*0.12*	*-0.48*	*-0.48*
	Maximum		*85.7*	*84.5*	*84.6*	*85.6*	*2.33*	*5.48*	*5.72*
	Average		*84.6*	*83.7*	*82.3*	*82.4*	*1.28*	*2.89*	*2.91*
KL	0	49	84.5	83.1	80.3	80.3	1.66	4.97	4.97
	10	45	84.4	83.3	81.3	81.0	1.3	3.67	4.03
	20	40	84.4	83.2	81.5	81.3	1.42	3.44	3.67
	30	35	84.3	83.4	81.4	81.2	1.07	3.44	3.68
	40	30	85	84.2	82.5	82.2	0.94	2.94	3.29
	50	25	84.7	84.0	81.9	82.1	0.83	3.31	3.07
	60	20	85.1	83.9	82.8	82.7	1.41	2.7	2.82
	70	15	83.8	82.5	82.3	82.4	1.55	1.79	1.67
	80	10	83.9	82.1	81.1	79.4	2.15	3.34	5.36
	90	5	82.5	81.7	78.7	78.7	0.97	4.61	4.61
	Minimum		*82.5*	*81.7*	*78.7*	*78.7*	*0.83*	*1.79*	*1.67*
	Maximum		*85.1*	*84.2*	*82.8*	*82.7*	*2.15*	*4.97*	*5.36*
	Average		*84.3*	*83.1*	*81.4*	*81.1*	*1.36*	*3.41*	*3.68*
PCA	0	49	79.6	78.9	77.8	78.5	0.88	2.26	1.38
	10	45	83.7	82.2	78.5	79.0	1.79	6.21	5.62
	20	40	82.9	81.7	79.7	80.2	1.45	3.86	3.26
	30	35	83.3	82.8	80.8	80.9	0.6	3	2.88
	40	30	83.7	82.3	82.1	82.3	1.67	1.91	1.67
	50	25	81.8	81.6	70.2	70.4	0.24	14.18	13.94
	60	20	83.3	82.5	82.3	82.3	0.96	1.2	1.2
	70	15	82.2	80.8	81.5	81.4	1.7	0.85	0.97
	80	10	83	81.7	81.1	81.0	1.57	2.29	2.41
	90	5	75.7	74.2	75.5	75.3	1.98	0.26	0.53
	Minimum		*75.7*	*74.2*	*70.2*	*70.4*	*0.24*	*0.26*	*0.53*
	Maximum		*83.7*	*82.8*	*82.3*	*82.3*	*1.98*	*14.18*	*13.94*
	Average		*81.9*	*80.9*	*79.0*	*79.1*	*1.26*	*4.21*	*4.03*

Table 4. Comparison of number of SVs

Feature Reduction	Feature Reduction	LMSVM Data	Number of SV	SV Reduction	Accuracy

Algorithm	(%)	Reduction (%)	SVM	LMSVM	LMSVM vs SVM (%)	Improvement (LMSVM – SVM)
CA	0	7.10	639	586	8.22	0.00
	10	3.80	598	570	4.72	0.20
	20	3.97	574	545	5.05	0.10
	30	3.37	533	509	4.60	0.10
	40	0.70	500	494	1.15	-0.10
	50	0.27	458	456	0.49	-0.10
	60	5.47	409	381	6.97	-0.20
	70	0.00	364	364	0.00	0.00
	80	0.00	361	361	0.00	0.00
	90	0.00	366	366	0.00	0.00
KL	0	7.10	639	586	8.22	0.00
	10	5.60	605	563	6.86	0.30
	20	7.90	561	505	9.98	0.20
	30	3.53	544	519	4.55	0.20
	40	1.17	476	467	1.79	0.30
	50	8.93	530	477	10.05	-0.20
	60	0.63	392	388	1.08	0.10
	70	2.33	418	404	3.35	-0.10
	80	3.40	476	450	0.00	1.70
	90	1.90	398	389	2.26	0.00
PCA	0	16.83	669	548	18.19	-0.70
	10	7.83	669	612	8.49	-0.50
	20	19.93	660	517	21.60	-0.50
	30	16.87	633	514	18.80	-0.10
	40	2.10	593	578	2.57	-0.20
	50	2.07	632	616	2.53	-0.20
	60	0.73	455	450	1.15	0.00
	70	1.03	426	420	1.41	0.10
	80	10.53	376	347	7.91	0.10
	90	0.83	414	412	0.66	0.20

4.4. Computational complexity

Although CA-based classifiers generated better recognition rates than SVM-based classifiers, their models are significantly more complex than their SVM-based counterparts. The computational complexity of these classifiers is the subject of this subsection. Before delving into the derivation, the adopted nomenclature is summarized in Table 5.

As previously mentioned, the adopted CA is formed of 6 layers, with 2000, 1000, 500, 250, 125 and 63 columns in the consecutive layers; and each column contains $M = 20$ neurons. Each column in layer i has a weight matrix of size $M \times L_i$. The total number of entries in CA*'s weight matrix is computed using (6) and found to be 106,641,880 weights. However, a small portion of these weights are usually non-zero and belong to the matrices of the firing columns. Therefore, a sparse matrix can be saved with some overhead information to keep track of firings columns to predict the class of a new instance. The total number of firing columns in the network is computed using (7). To simplify the computations, a uniform distribution of firing columns across layers will be assumed. Therefore, the number of firing columns per layer is computed in (8). Finally, the total number of non-zero weights can be approximated by (9).

$$N_W = \sum_{i=1}^{l} M \times {L_i}^2 \quad (6)$$

$$N_{FC} = \gamma \times \sum_{i=1}^{l} L_i \quad (7)$$

$$N_{FC/L} = \frac{N_{FC}}{l} \quad (8)$$

$$N_{NZW} = \sum_{j=1}^{l} M \times L_j \times N_{FC/L} \quad (9)$$

Using an SVM model, a new point is assigned to a class using (10) which involves computing the kernel value in (11) which requires F multiplications and $(\mathrm{F}-1)$ additions. Depending on the adopted kernel, a constant number of additional operations will be performed; RBF kernel requires the calculation of an exponential, estimated by its Taylor series expansion in as shown in (12), with the number of terms $\mathrm{n} \leq 10$ (Moller 2011), equivalent to approximately 70 operations. These operations can be significant compared to the number of features and will be represented by the variable c. Therefore, $\mathrm{O}(\mathrm{N_{SV}}(\mathrm{F}+\mathrm{c}))$ operations are required for each new point during online prediction. LMSVM has the same order of operations but has a smaller SV set cardinality. Clearly, SVM based classifiers have a much simpler decision rule which requires significantly less memory requirements and computational resources, leading to lower power consumption compared to CA. For sports betting scenarios where decision need to be made in real time and partial information is better than no information at all, adopting a less computationally expensive solution may be worth the slight reduction in accuracy. Depending on the specific scenario and the importance of accuracy over speed, an appropriate algorithm can be selected from the proposed approached. For example, if the monetary value of the bet is very large, users may want to invest in a computationally expensive algorithm to improve their decision making.

$$y_p = \sum_{i=1}^{N_{SV}} \alpha_i K(x_{new}, x_i) \quad (10)$$

$$K(x_1, x_2) = \exp\left(-\frac{|x_1 - x_2|^2}{\sigma^2}\right) \quad (11)$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!} + \cdots = \sum_{i=0}^{\infty} \frac{x^i}{i!} \quad (12)$$

Table 6 summarizes the empirical computational complexity of the prediction models produced by CA, SVM and LMSVM, calculated using the results in Tables 4 and 7 using (5) through (11). As shown in Table 7, the firing rate is between 20% and 35% which reduces the number of non-zero weights to the order of 10^7, significantly higher than SVM whose requirements are in the order of 10^4. However, this cost increase was mitigated by the increase in performance.

Table 5: Nomenclature

N_{SV}	support vectors set cardinality
$K(x_i, x_j)$	kernel value for instances x_i and x_j
F	feature vector length
c	constant operations needed in computing kernel value
α	weight of SV
x_{new}	new point to classify
x_i	support vector i
y_p	predicted class of x_{new}
l	number of layers in CA network
M_j	size of column j

L_i: number of columns in layer i
γ: firing rate
N_{FC}: number of firing columns in the network
$N_{FC/L}$: number of firing columns per layer
N_{NZW}: number of non-zero weights
N_W: total number of weights in the network

Table 6: Empirical computational complexity

Feature Reduction Algorithm	Feature Reduction (%)	Approximate Number of Operations			Relative Cost		
		CA*	SVM	LMSVM	CA*/SVM	CA*/LMSV	SVM/LMSVM
CA	0	12,819,814	76680	70320	167	182	1.09
	10	12,923,200	68770	65550	188	197	1.05
	20	14,370,598	63140	59950	228	240	1.05
	30	16,541,696	55965	53445	296	310	1.05
	40	17,110,316	50000	49400	342	346	1.01
	50	17,368,780	43510	43320	399	401	1.00
	60	18,092,480	36810	34290	492	528	1.07
	70	17,575,552	30940	30940	568	568	1.00
	80	17,317,088	28880	28880	600	600	1.00
	90	16,541,696	27450	27450	603	603	1.00
KL	0	12,406,272	76680	70320	162	176	1.09
	10	13,026,585	69575	64745	187	201	1.07
	20	13,853,670	61710	55550	224	249	1.11
	30	14,990,912	57120	54495	262	275	1.05
	40	16,490,003	47600	46700	346	353	1.02
	50	17,265,395	50350	45315	343	381	1.11
	60	18,144,172	35280	34920	514	520	1.01
	70	17,834,016	35530	34340	502	519	1.03
	80	17,213,702	38080	36000	452	478	1.06
	90	16,541,696	29850	29175	554	567	1.02
PCA	0	12,147,808	80280	65760	151	185	1.22
	10	12,509,657	76935	70380	163	178	1.09
	20	13,698,592	72600	56870	189	241	1.28
	30	14,370,598	66465	53970	216	266	1.23
	40	15,662,918	59300	57800	264	271	1.03
	50	16,334,924	60040	58520	272	279	1.03
	60	15,507,840	40950	40500	379	383	1.01
	70	15,197,683	36210	35700	420	426	1.01
	80	14,629,062	30080	27760	486	527	1.08
	90	14,473,984	31050	30900	466	468	1.00

Table 7: Firing rate for CA* network

Feature Reduction Rate (%)	Feature Reduction Algorithm		
	CA	KL	PCA
0	24.8	24.0	23.5
10	25.0	25.2	24.2

20	27.8	26.8	26.5
30	32.0	29.0	27.8
40	33.1	31.9	30.3
50	33.6	33.4	31.6
60	35.0	35.1	30.0
70	34.0	34.5	29.4
80	33.5	33.3	28.3
90	32.0	32.0	28.0

Table 8 compares the empirical running times of the various feature reduction and classification algorithms considered in this paper. On average, CA requires about 150 times more training time than SVM and LMSVM for the full data, this number increasing to about 1000 for the data reduced at 90%. In terms of testing time, CA is around 90 times slower than SVM or LMSVM primarily due to the process of input propagation required for every instance, whereas SVM only implements a linear operation (equivalent to the decision function). Though computational complexity is a major concern, hardware and software solutions can greatly help improve this shortcoming through parallel processing. Furthermore, CA's testing time per instance remains within real time processing limits, while SVM and LMSVM are more suitable for energy aware environments. Concerning feature reduction algorithms, CA's clear advantage in performance is also countered by its high complexity ranging at over 10^4 times more than KL or PCA. A main difference between these algorithms is that CA requires training till convergence when eliminating less significant features, while KL and PCA compute statistical measures of features to rank attributes.

Furthermore, Table 9 shows a statistical comparison of the performance of CA, SVM and LMSVM using CA and KL for feature reduction. The relative performance of the mentioned classifiers in terms of accuracy and training time is shown. For example, SVM witnesses an average improvement of 0.73 in accuracy when CA is used for feature reduction (versus KL), while a maximum degradation of about 6 seconds in training time is seen by LMSVM when CA is used for feature reduction. Overall, all three classifiers see an improvement in performance when coupled with CA at the expense of a degradation in training time.

Table 8: Running Times (in seconds) for Feature Reduction and Classification Algorithms

Feature Reduction Algorithm	Feature Reduction Rate (%)	Feature Reduction Running Time	Training Time			Testing Time		
			CA	SVM	LMSVM	CA	SVM	LMSVM
CA	0	3026	5004	29.9326	26.7867	0.7807	0.0055	0.0058
	10	1328	1632	0.0519	0.0737	0.8940	0.0097	0.0090
	20	827	1622	0.0377	0.0589	0.2006	0.0088	0.0084
	30	382	1268	0.0491	0.0591	0.8747	0.0150	0.0097
	40	114	1036	0.0463	0.0597	0.6881	0.0132	0.0079
	50	111	867	0.0332	0.0575	0.4577	0.0071	0.0065
	60	111	624	0.0326	0.0519	0.1923	0.0048	0.0044
	70	85	608	0.0194	0.0510	0.5456	0.0038	0.0037
	80	34	506	0.0136	0.0487	0.1604	0.0048	0.0031
	90	17	473	0.0121	0.0425	0.7361	0.0029	0.0027
KL	0	0.1584	3798	23.4726	20.8667	0.8355	0.0064	0.0053

	10	0.1670	1913	0.0451	0.0596	0.9183	0.0088	0.0083
	20	0.0052	668	0.0333	0.0444	0.0222	0.0075	0.0066
	30	0.0011	212	0.0312	0.0459	0.1755	0.0080	0.0076
	40	0.0402	80	0.0270	0.0512	0.4577	0.0070	0.0069
	50	0.0026	38	0.0218	0.0430	0.8779	0.0063	0.0050
	60	0.0056	31	0.0244	0.0486	0.1496	0.0043	0.0043
	70	0.0001	12	0.0179	0.0340	0.0354	0.0036	0.0033
	80	0.0011	4	4.0071e-04	0.0368	0.5867	2.5616e-04	1.3591e-04
	90	0.0039	2	0.0090	0.0422	0.5802	0.0023	0.0026
PCA	0	0.1656	690	4.5590	4.3778	0.8977	0.0094	0.0072
	10	0.0827	280	0.0330	0.0567	0.8485	0.0078	0.0076
	20	0.007	256	0.0348	0.0493	0.1966	0.0091	0.0077
	30	0.0032	228	0.0269	0.0476	0.2221	0.0069	0.0060
	40	0.0197	125	0.0267	0.0541	0.2018	0.0070	0.0065
	50	0.0024	109	0.0471	0.0801	0.9172	0.0057	0.0055
	60	0.0024	47	0.0223	0.0586	0.3836	0.0047	0.0051
	70	0.0018	20	0.0176	0.0398	0.3484	0.0040	0.0041
	80	0.0015	12	0.0138	0.0284	0.8369	0.0031	0.0026
	90	0.0019	9	0.0118	0.0427	0.7756	0.0025	0.0027

Table 9: Statistical Comparison of CA vs KL for Feature Reduction

Statistic	Performance			Training Time		
	CA	SVM	LMSVM	CA	SVM	LMSVM
Minimum	-0.3	-0.5	-0.2	-281	0.0015	0.0003
Maximum	0.8	4.3	4.3	1206	6.4600	5.9200
Average	0.400	0.730	0.980	688	0.6546	0.6017

5. Conclusion

In this paper, an affective computing framework to classify sports articles as objective or subjective was presented. Articles from webpages were extracted, parsed, and tagged with POS. Feature reduction using CA* which is trained with a weighted entropy weight update rule and a CE cost function was applied to extracted semantic and syntactic features. Several classifiers were tested including SVM, LMSVM and CA*. Feature reduction improved accuracy, by eliminating approximately 40% of the features. Furthermore, CA* produced the best accuracy, equal to 85.6%, among all classifiers and improving on other methods by almost 3%. While this accuracy gain may seem minor, it might make a big difference in high stakes betting where large amounts of money are involved. However, the decision rule produced by CA* is very complex and can make real time computations on devices with limited computational capabilities impossible. Therefore, adopting another classifier like SVM would produce a simpler hyper-plane representation but slightly decreases the prediction accuracy. For applications where a decrease in accuracy can be tolerated, using CA* for feature reduction followed by a SVM for classification can be an attractive alternative to achieve real time performance and decrease computational requirements. Even though this work focused on sports articles and included features commonly encountered in this domain (e.g. numbers), the approach can be extended to other domains such as medical document analysis, stock market investments and political speech analysis domains where the minor gains in accuracy may have an even larger impact that in sports betting, and will be investigated in future work.

Acknowledgements

This work was partially funded by Intel and the University Research Board at the American University of Beirut. We would also like to acknowledge the help of Professor Lina Choueiri from the Department of English at the American University of Beirut.

References

Abbasi, A., France, S., Zhang, Z., & Chen, H. (2011). Selecting attributes for sentiment classification using feature relation networks. In IEEE Transactions on Knowledge and Data Engineering, 23, 447-462.

Atyabi, A., Luerssen, M., Fitzgibbon, S., & Powers, D. M. (2012, June). Evolutionary feature selection and electrode reduction for EEG classification. In IEEE Congress on Evolutionary Computation (CEC2012), 1-8.

Benamara, F., Cesarano, C., Picariello, A., Reforgiato, D., & Subrahmanian, V.S. (2007). Sentiment analysis: Adjectives and adverbs are better than adjectives alone. In Proc. of the Int. Conf. on Weblogs and Social Media.

Bian, W., & Tao, D. (2011). Max-min distance analysis by using sequential SDP relaxation for dimension reduction. In IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(5), 1037-1050.

Bi, J., Bennett, K., Embrechts, M., Breneman, C., & Song, M. (2003). Dimensionality reduction via sparse support vector machines. In the Journal of Machine Learning Research, 3, 1229-1243.

Calais Guerra, P. H., Veloso, A., Meira Jr, W., & Almeida, V. (2011, August). From bias to opinion: a transfer-learning approach to real-time sentiment analysis. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 150-158.

Chang, C.C., & Lin, C.J. (2011). LIBSVM: a library for support vector machines. In ACM Transactions on Intelligent Systems and Technology, 2, 27.

Chen, Y., Miao, D., Wang, R., & Wu, K. (2011). A rough set approach to feature selection based on power set tree. In Knowledge-Based Systems, 24(2), 275-281.

Chua, F. C. T. (2009). Dimensionality reduction and clustering of text documents. Singapore Management University.

Das, A., & Bandyopadhyay, S. (2010, August). Subjectivity detection using genetic algorithm. In the 1st Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA10), Lisbon, Portugal, August.

Deepthi, D. R., Krishna, G. R., & Eswaran, K. (2007). Automatic pattern classification by unsupervised learning using dimensionality reduction of data with mirroring neural networks. In arXiv preprint arXiv:0712.0938.

Devitt, A., & Ahmad, K. (2007). Sentiment analysis in financial news: A cohesion-based approach. In Proc. of the Association for Computational Linguistics, 984–991.

Edelman, G. M., & Mountcastle, V. B. (1982). The Mindful Brain. The MIT Press.

Esuli, A., & Sebastiani, F. (2006). SentiWordNet: A publicly available lexical resource for opinion mining. In Proc. of Language Resources and Evaluation, 417-422.

Formisano, E., De Martino, F., Bonte, M., & Goebel, R. (2008). " Who" Is Saying" What"? Brain-Based Decoding of Human Voice and Speech. In Science, 322(5903), 970-973.

Godbole,N., Srinivasaiah, M., & Skiena, S. (2007). Large-scale sentiment analysis for news and blog. In Proc. of the Int. Conf. on Weblogs and Social Media, 219-222.

Hajj N., & Awad, M. (2013, August). Weighted Entropy Cortical Algorithms for Modern Standard Arabic Speech Recognition. In International Joint Conference on Neural Networks (IJCNN), Dallas, TX.

Han, Y., & Yu, L. (2012). A variance reduction framework for stable feature selection. In Statistical Analysis and Data Mining, 5(5), 428-445.

Hashmi, A. G., & Lipasti, M. H. (2009). Cortical Columns: Building Blocks for Intelligent Systems. In IEEE Symposium on Computational Intelligence for Multimedia Signal and Vision Processing, 21 - 28.

Hashmi, A. G., & Lipasti, M. H. (2010). Discovering Cortical Algorithms. In Proceedings of the International Conference on Fuzzy Computation and International Conference on Neural Computation, Valencia, Spain, 196-204.

Heerschop, B., Hogenboom, A., & Frasincar, F. (2011a). Sentiment Lexicon Creation from Lexical Resources. In 14th Int. Conf. on Business Information Systems, 87, 185–196.

Heerschop, B., Van Iterson, P., Hogenboom, A., Frasincar, F., & Kaymak, U. (2011b). Analyzing Sentiment in a Large Set of Web Data while Accounting for Negation. In Advances in Intelligent Web Mastering–3, 195-205.

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. In Science, 313(5786), 504-507.

Moller, C. (2011). Experiments with MATLAB. The MathWorks, Co.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs Up? Sentiment Classifcation using Machine Learning Techniques. In Proc. Empirical Methods in Natural Language Processing, Philadelphia, 79-86.

Perantonis, S. J., & Virvilis, V. (1999). Input feature extraction for multilayered perceptrons using supervised principal component analysis. In Neural processing letters, 10(3), 243-252.

Raymer, M. L., Punch, W. F., Goodman, E. D., Kuhn, L. A., & Jain, A. K. (2000). Dimensionality reduction using genetic algorithms. In IEEE Transactions on Evolutionary Computation, 4(2), 164-171.

Rizk, Y., & Awad, M. (2012, August). Syntactic Genetic Algorithm for a Subjectivity Analysis of Sports Articles. In 11th IEEE International Conference on Cybernetic Intelligent Systems, Limerick, Ireland.

Rizk, Y., Mitri, N., & Awad, M. (2013, August). A Local Mixture Based SVM for an Efficient Supervised Binary Classification. In International Joint Conference on Neural Networks, Dallas, TX.

Shafiei, M., Wang, S., Zhang, R., Milios, E., Tang, B., Tougas, J., & Spiteri, R. (2007, April). Document representation and dimension reduction for text clustering. In Data Engineering Workshop, 2007 IEEE 23rd International Conference on (pp. 770-779). IEEE.

Silva, L. M., Marques de Sá, J., & Alexandre, L. A. (2005, April). Neural network classification using Shannon's entropy. In ESANN, 217-222.

Silva, L. M., Marques de Sá, J., & Alexandre, L. A. (2008). Data classification with multilayer perceptrons using a generalized error function. In Neural Networks, 21(9), 1302-1310.

Tang, E. K., Suganthan, P. N., Yao, X., & Qin, A. K. (2005). Linear dimensionality reduction using relevance weighted LDA. In Pattern recognition, 38(4), 485-493.

Toutanova, K., Klein, D., Manning, C. D., & Singer, Y. (2003). Feature-Rich Part-Of-Speech Tagging with a Cyclic Dependency Network. In NAACL'03: Proc. of the 2003 Conf. of the North American Chapter of the Association of Computational Linguistics on Human Language Technology, Edmonton, Canada, 173-180.

Wang, M., Sha, F., & Jordan, M. I. (2010). Unsupervised kernel dimension reduction. In Advances in Neural Information Processing Systems, 2379-2387.

Wang, H., Can, D., Kazemzadeh, A., Bar, F., & Narayanan, S. (2012, July). A system for real-time twitter sentiment analysis of 2012 us presidential election cycle. In Proceedings of the ACL 2012 System Demonstrations, pp. 115-120.

Wiebe, J., & Riloff, E. (2011, December). Finding Mutual Benefit between Subjectivity Analysis and Information Extraction. In IEEE Transactions on Affective Computing, 1.

Wiebe, J., Wilson, T., Bruce, R., Bell, M., & Martin, M. (2002). Learning subjective language. In Technical Report TR-02-100, Department of Computer Science, University of Pittsburgh, Pittsburgh, Pennsylvania.

Wilson, T., Wiebe, J., & Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In Proc. of the Human Language Technology Conf. and the Conf. on Empirical Methods in Natural Language Processing, 347–354.

Yang, Y., & Pedersen, J. O. (1997, July). A comparative study on feature selection in text categorization. In ICML (Vol. 97, pp. 412-420).

Yu, H., & Hatzivassiloglou, V. (2003). Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing, 129-136.