Language comprehension as a multi-label classification problem

Konstantin Sering (corresponding author: konstantin.sering@uni-tuebingen.de), Petar Milin, R. Harald Baayen

January 16, 2018

The initial stage of language comprehension is a multi-label classification problem. Listeners or readers, presented with an utterance, need to discriminate between the intended words and the tens of thousands of other words they know. We propose to address this problem by pairing a network trained with the learning rule of Rescorla and Wagner (1972) with a second network trained independently with the learning rule of Widrow and Hoff (1960). The first network has to recover from sublexical input features the meanings encoded in the language signal, resulting in a vector of activations over the lexicon. The second network takes this vector as input and further reduces uncertainty about the intended message. Classification performance for a lexicon with 52,000 entries is good. The model also correctly predicts several aspects of human language comprehension. By rejecting the traditional linguistic assumption that language is a (de)compositional system, and by instead espousing a discriminative approach (Ramscar, 2013), a more parsimonious yet highly effective functional characterization of the initial stage of language comprehension is obtained.

multi-label classification, language comprehension, error-driven learning, Rescorla-Wagner, Widrow-Hoff

1 Introduction

Table 1 presents 10 simple sentences. When reading these sentences, the letters and their combinations succeed in bringing to the fore a small number of meanings while dismissing thousands of others as irrelevant. Each of the sentences encodes a small number of a much larger set of meanings. Therefore the sentences present the reader with a multi-label classification problem.

In this paper we model this problem as follows. First, we represent the orthographic input by means of letter trigrams. For the first sentence, these are `#Ma Mar ary ry# y#p #pa pas ass sse sed ed# d#a #aw awa way ay#` (the `#` symbol represents the

space character). Letter trigrams provide a much richer and denser representation of the visual input than do orthographic words. For the data in Table 1, there are $n = 104$ distinct letter trigrams, to which we refer as cues.

The second column lists the lexical "meanings", or more precisely, the lexomes, that are expressed in the sentences. These lexomes are the targets of multi-label classification. Lexomes are pointers to locations in a high-dimensional semantic vector space (defined below). Note that past-tense word forms such as *passed* (regular) and *ate* (irregular) are coupled with the lexomes PASS and EAT as well as with the lexome for the past tense, PAST. Furthermore, the two word forms *apple* and *pie* are coupled with one lexome APPLEPIE, and the three expressions with the word forms *kicked the bucket*, *passed away*, and *died*, are all linked with the same lexome DIE. As will become clear below, lexomes are placeholders for (or pointers to) meanings that themselves are formally represented in a vector space defined by a second network.

Table 1: Sentences, lexomes in the message, and frequency of occurrence (F). The total number of learning events is $k = 771$.

	Sentence	Lexomes in the message	F
1	Mary passed away	MARY DIE PAST	40
2	Bill kicked the ball	BILL KICK PAST DEF BALL	100
3	John kicked the ball away	JOHN KICK PAST DEF BALL AWAY	120
4	Mary died	MARY DIE PAST	300
5	Mary bought clothes for the ball	MARY BUY PAST CLOTHES FOR DANCEPARTY	20
6	Ann bought a ball	ANN BUY PAST INDEF BALL	45
7	John filled the bucket	JOHN FILL PAST DEF BUCKET	100
8	John kicked the bucket	JOHN DIE PAST	10
9	Bill ate the apple pie	BILL EAT DEF APPLEPIE	3
10	Ann tasted an apple	ANN TASTE PAST INDEF APPLE	33

Is it possible to discriminate between the targeted "meanings" (lexomes) given the letter trigrams in the sentences? We will show that considerable headway can be made by an error-driven incremental multi-label classifier that comprises two simple networks, each with only an input layer and an output layer. In what follows, we first provide a formal definition of the algorithm, and illustrate it for the sentences in Table 1, our working example. We then turn to a more realistic example in which lexomes targeted in around a million of utterances have to be discriminated from some 52,000 other lexomes.

2 An algorithm for multi-label classification

The problem of incremental learning of multi-label classification is defined by a sequence of events at which a set of features (henceforth cues) are present and generate predictions about classes (henceforth outcomes), only some of which are actually present in the learning event. The mismatch between predicted outcomes and the outcomes which are actually present in a learning event provides the error driving learning.

From a total of n distinct cues and m possible outcomes, only small subsets will be present in a given learning event. Let k denote the number of learning events (learning events may repeat, cf. *good morning* and *tickets please*). The learning events are presented in a specific order, which has an influence on the actual estimates of the algorithm. This is a desired property and built in on purpose as things that have been experienced in the near past should leave a stronger impact on the estimates than those which occurred a long time ago.

The classification problem can be phrased as learning the association between the cues and the outcomes. Here we restrict these associations to be linear and additive between the different cues. Equation 1 depicts this linear association in matrix notation. The target labels in the event are represented by $\mathbf{t}$, a binary vector of zeros and ones, of length m. Each dimension in this vector corresponds to one unique outcome, such that $\mathbf{t}$ is 1 in all those dimensions where outcome lexomes are present in the event and 0 otherwise. The cues are represented by a binary vector $\mathbf{c}$ of length n that is 1 for the cues present in the event and 0 otherwise. Each dimension in $\mathbf{c}$ corresponds to one unique cue. We refer to W as the weight matrix as it linearly transforms, as far as possible, the cue vector to the target or outcome vector. Therefore $\mathbf{W}$ has the dimensions $m \times n$. The weight matrix is not determined a priori but has to be estimated:

$$\mathbf{t} = \mathbf{W}\mathbf{c}. \tag{1}$$

For any specific learning event, equation (1) usually can be solved perfectly, if we would allow for a different $\mathbf{W}$ for each learning event. Usually, one either demands to have a fixed $\mathbf{W}$ over all learning events and estimates the best $\mathbf{W}$ under this constraint, or one implements a learning rule which allows to update $\mathbf{W}$ deterministically from learning event to learning event. For the modeling of human lexical learning, stepwise updating is preferred.

One of the learning rules that shows a good match to human behaviour is the learning rule suggested by Rescorla and Wagner (1972). It implements a two-step approach where the learner first predicts the outcomes with the current weights and then in a second step updates the efferent weights from the cues that appeared in the learning event according to the error or mismatch of the predictions.

Predictions are calculated by summing over all the association weights between present cues and all known outcomes. The resulting activation vector $\mathbf{a}$ is of length m, with dimensions corresponding to the different outcomes:

$$\mathbf{a} = \mathbf{W}\mathbf{c}. \tag{2}$$

The real-valued activation vector $\mathbf{a}$ can be compared to the actual outcomes in the binary target vector $\mathbf{t}$. The difference between these two is the error. The algorithm updates the weights in the weight matrix representing the association strengths between the cues present in the input and a given outcome j with a proportion η of the error:

$$\Delta w_{ij} = \begin{cases} 0 & \text{if cue } c_i \text{ is absent} \\ \eta(t_j - a_j) & \text{if cue } c_i \text{ is present.} \end{cases} \tag{3}$$

Calculating the elementwise difference in equation (3) can be summarized in matrix notation because of the binary nature of the cue vector $\mathbf{c}$. The update $\Delta\mathbf{W}$ can be calculates as:

$$\Delta\mathbf{W} = \eta(\mathbf{t} - \mathbf{a})\mathbf{c}^T, \tag{4}$$

Note that we use a matrix product between the difference vector $(\mathbf{t} - \mathbf{a})$ and the cue vector $\mathbf{c}^T$, therefore the update $\Delta\mathbf{W}$ is a $m \times n$ matrix. The actual updating from learning event $i-1$ to learning event i requires adding the update to the weight matrix:

$$\mathbf{W}^i = \mathbf{W}^{i-1} + \Delta\mathbf{W}. \tag{5}$$

The learning rule described by Rescorla and Wagner includes specific parameters α_i that relax the asssumption that all cues are equally salient, and parameters β_j that weight the error differently depending on whether a given outcome is present or absent. The learning rate η is identical to $\alpha \times \beta$ assuming all alphas are the same ($\alpha = \alpha_i$) and both betas are the same ($\beta = \beta_1 = \beta_2$). As a consequence, our model has η as its only free parameter. In practice, η itself is not fitted, but is set to 0.01 or 0.001 depending on the number of cues n and outcomes m and the number of learning events k.

Rescorla-Wagner learning is independent in its outcomes. This is beneficial computationally as we can calculate the rows of the weight matrix in parallel. It is also convenient for actual modeling, as it allows us to consider subsets of outcomes. Given the same cues and the same learning events, the model will generate exactly the same predictions for a single outcome regardless of what other outcomes are included in the computation. As the number of outcomes in large text corpora can be huge (hundreds of thousands, or even millions of different outcomes can be at issue), it is convenient to be able to zoom in on subsets of outcomes without affecting the results. The same is not true for the cues. If we choose to select a subset of cues, in general this will change all weights and results obtained will depend on the choice of this subset. On the downside, the independence of the outcomes implies that the network is blind to commonalities between outcomes. Because it is completely agnostic about the number of outcomes in the world, it cannot benefit from any knowledge pertaining to the distribution of outcomes.

In this study, we propose a second network, which is stacked on the first network, but is trained independently. This second network takes the activations $\mathbf{a}$ of length m of the first network as input and maps these linearly on the expected outcomes $\mathbf{t}$. This linear mapping can be expressed as a decision matrix $\mathbf{D}$ of dimensions m by m.

$$\mathbf{t} = \mathbf{D}\mathbf{a}. \tag{6}$$

We apply the same learning algorithm as used for the first network, with the notable difference that the cues are no longer associated with a binary vector, but with the real-valued vector of activations produced on the output layer of the first network. As before, we first calculate the predictions $\mathbf{p}$ over the outcomes:

$$\mathbf{p} = \mathbf{D}\mathbf{a}. \tag{7}$$

The error in the second network is the difference between the binary target vector $\mathbf{t}$ and the real valued prediction vector $\mathbf{p}$ weighted by its activation $\mathbf{a}$. The update of the

decision matrix $\mathbf{\Delta D}$, with learning rate η_D is shown in equation (8) and has the same form as equation (4) in the first network. The notable difference is that now the error is not filtered with a binary vector but weighted by the real valued vector $\mathbf{a}$.

$$\Delta\mathbf{D} = \eta_D(\mathbf{t} - \mathbf{p}) \cdot \mathbf{a}^T \tag{8}$$

The update $\mathbf{\Delta D}$ from learning event $i-1$ to learning event i is added to the decision matrix:

$$\mathbf{D}^i = \mathbf{D}^{i-1} + \Delta\mathbf{D}. \tag{9}$$

This way of training the network is equivalent to the learning rule of Widrow and Hoff (1960). Thus, a learning rule from physics and engineering, and an independently developed learning rule from animal learning in psychology, are essentially the same and differ only in that the latter takes a binary-valued input vector and the former a real-valued input vector. As the number of cues in a learning event tends to be quite small compared to the total number of cues, the updating of weights of the first network is effectively restricted to the efferent weights from the few cues that are present. Updating the second network involves adjusting all weights, which renders the training of the second network computationally much more expensive.

The weight matrix $\mathbf{W}$ is initialized with zeros and the decision matrix $\mathbf{D}$ is initialized with ones on the main diagonal elements and is set to zero elsewhere. Initializing $\mathbf{D}$ with the identity matrix enables this network to take what the first network has learned as its starting point. It is worth noting that the target vector $\mathbf{t}$ and the cue vector $\mathbf{c}$ are usually sparse. The independence of outcomes in the first network depends on the initialisation of the weight matrix $\mathbf{W}$ with zeros. For any other initialisation the independence does not hold true.

As learning of $\mathbf{D}$ depends on the actual values in the weight matrix $\mathbf{W}$, when training the two networks we have three options. The first option is to run through all learning events and learn $\mathbf{W}$ and then to run through all the learning events again to learn $\mathbf{D}$ with the fixed weight matrix $\mathbf{W}$ resulting from the first run:

$$\Delta\mathbf{D}^i = (\mathbf{t}^i - \mathbf{D}^{i-1}\mathbf{W}^k\mathbf{c}^i) \cdot (\mathbf{W}^k\mathbf{c}^i)^T. \tag{10}$$

The second option is to use an interleaved approach where for each learning event we first update the weight matrix $\mathbf{W}$ and then with this updated weight matrix update the decision matrix $\mathbf{D}$:

$$\Delta\mathbf{D}^i = (\mathbf{t}^i - \mathbf{D}^{i-1}\mathbf{W}^i\mathbf{c}^i) \cdot (\mathbf{W}^i\mathbf{c}^i)^T. \tag{11}$$

As a third option, we can update the decision matrix and the weight matrix interleaved, but always using the weight matrix of the preceding event:

$$\Delta\mathbf{D}^i = (\mathbf{t}^i - \mathbf{D}^{i-1}\mathbf{W}^{i-1}\mathbf{c}^i) \cdot (\mathbf{W}^{i-1}\mathbf{c}^i)^T. \tag{12}$$

Cognitively, only the second and third options are realistic as models of incremental learning.

Note that the two networks use the error independently: the target vector $\mathbf{t}$ appears both in equation (4) and in equation (8). There is no backpropagation of error across the two networks.

2.1 Computational shortcuts for large data sets

The above algorithm works well for small numbers of cues, outcomes and learning events. For the first network an efficient implementation exists which scales up to 100,000 cues and 100,000 outcomes and several millions to billions of learning events. The implementation is written in Python and implements the described incremental learning of the weight matrix $\mathbf{W}$ and is available as free software at `https://github.com/quantling/pyndl` (Sering et al., 2018). The implementation heavily exploits the fact that the cue vector and the outcome vector are binary vectors with only a few cues present in each learning event.

For the decision matrix $\mathbf{D}$, we do not know of any efficient algorithm that scales up to 50,000 to 100,000 outcomes. Work is in progress to develop an efficient implementation, and the possibility of using the Kalman filter (Kalman, 1960) instead of the Widrow-Hoff learning rule is simultaneously being explored. For the time being, we therefore fall back on a regression like approximation of the decision matrix. This approximation implements option 1 with equation (10). In a first step, we calculate the final weights matrix $\mathbf{W}^{\text{end}}$ by going once through all the learning events with the efficient algorithm for Rescorla-Wagner learning. In the second step, we stack all the cue vectors column-wise to an $n \times k$ cue matrix $\mathbf{C}$ and left multiply the cue matrix with the final weight matrix $\mathbf{W}^{\text{end}}$. This step effectively calculates the activations of all the learning events with respect to the final weight matrix. In the third step, we need to solve

$$\mathbf{T} = \mathbf{D}\mathbf{W}^{\text{end}}\mathbf{C} \tag{13}$$

for the decision matrix $\mathbf{D}$. Here the target matrix $\mathbf{T}$ denotes the column-wise stacked $m \times k$ matrix consisting of all the target vectors.

As the number of the learning events usually is much larger than the number of outcomes, i.e., $m \ll k$, when dealing with many learning events we do not want to solve equation (13) by calculating the generalized inverse of the $m \times k$ activation matrix $\mathbf{A} = \mathbf{W}^{\text{end}}\mathbf{C}$). Calculations are simplified by first right multiplying with the transpose of the activation matrix $\mathbf{A}^T$ and then calculating the inverse of the $m \times m$ matrix $(\mathbf{A}\mathbf{A}^T)$:

$$\begin{aligned} \mathbf{T} &= \mathbf{D}\mathbf{A} \\ \mathbf{T}\mathbf{A}^T &= \mathbf{D}\mathbf{A}\mathbf{A}^T \\ \mathbf{T}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} &= \mathbf{D}\mathbf{A}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} \\ \mathbf{T}\mathbf{A}^T(\mathbf{A}\mathbf{A}^T)^{-1} &= \mathbf{D}. \end{aligned} \tag{14}$$

Note that although the cue matrix $\mathbf{C}$ and the target matrix $\mathbf{T}$ are sparse binary matrices, the activation matrix $\mathbf{A}$ is dense. Furthermore, $(\mathbf{A}\mathbf{A}^T)$ for large m usually is nearly regular and therefore its inverse has to be calculated with the generalized inverse or similar algorithms ignoring eigenvalues below a predefined cutoff.

2.2 Evaluating model performance

We can base the prediction of the outcomes (or class labels) for each event, given the cues present in that event, in two ways. Baseline performance is assessed with the first

network only by using the activation **a** to predict outcomes. The performance of the two networks jointly is calculated from the prediction vector **p**. The closer the value for an outcome in the activation or prediction vector is to 1, the more the system believes that this outcome should be classified as present in the event. Concrete predictions can be generated in several ways.

A naive but simple way of gauging classification performance is to calculate the lowest empirical rank for the set of outcomes known to be encoded in an event, and to compare this rank to the cardinality of the set of outcomes present in the learning event. Ideally, the lowest empirical rank is identical to the cardinality. The more the two diverge, the more intruders are present.

As an example, the sentence "*everyone was quiet*" has the outcomes BE, QUIET and EVERYONE. The predictions of the models are BE, QUIET and EVERYONE, therefore no intruders are present and the lowest empirical rank is 3 which is the same as the cardinality of the three true outcomes. In contrary the sentence "*not so smart at all*" has the outcomes SO, NOT, ALL, AT and SMART but the worst rank of the predictions of one of these true outcomes is at rank 10 compared to the cardinality of 5. This is due to the presence of five intruders, namely BE, HEART, PART, SMALL and START. These examples show that in order to calculate the number of intruders it is necessary to know the true outcomes in advance.

More in general, it is desirable to predict outcomes without knowledge of what outcomes are actually encoded in the language input. Under this constraint, the simplest option is to set a fixed cutoff value so that every outcome with an activation or prediction that exceeds this cutoff value is classified as present. This option has the property that the classification of an outcome as present or absent is independent of the activation or prediction values of all the other outcomes. As learning events may have very different numbers of cues, due to very different sentence lengths, this method has the risk of incorrectly accepting as present more words for shorter sentences.

Another option is to sort the activation or prediction values by magnitude in decreasing order and to inspect the differences between consecutive values. A marked, abrupt drop in values, followed by a sequence of gradually and slowly diminishing values, if present, could be used as cutoff point, with only outcomes with values higher than this cutoff being classified as present. Another option is to generate an expected rank distribution of activations or predictions under random cue sampling for each given number of cues. For example, one could generate an expected rank distribution of activations for 10 cues by uniformly sampling 10 cues out of all possible cues. The 10 sampled cues are used to calculate the activations of all outcomes, which are then ordered by magnitude. This gives us one sample of a rank distribution under random cue sampling. If we generate 10,000 of such sample rank distributions, we can calculate the mean and the standard error of the activations at each rank. We call the mean activations at each rank the expected by-rank activation distribution. By comparing the observed activation at a given rank with its by-rank mean activation distribution for the event with the expected rank distribution, all outcomes with empirical scores higher than the expected rank distribution are classified as present.

Let $\{T\}_e$ denote the set of true outcomes for event e, and let $\{P\}_e$ denote the set of

predicted outcomes. For our data, the number of true outcomes $|\{T\}_e|$ is small compared to the total number of outcomes m. As a consequence, the number of correct rejections, i.e., the outcomes correctly classified as not being present, are not of interest. We consider here four performance metrics. The Hamming loss for an event e is the fraction of wrong labels, false positives f_e and misses m_e, to the total number of outcomes m in the model:

$$\text{Hamming} = \frac{f_e + m_e}{m} \tag{15}$$

A value closer to zero is better and a value of 0 means a perfect match. As for our data m is large, values of this statistic will all be close to zero. The Jaccard index is defined as the number of correctly predicted outcomes divided by the cardinality of the union of predicted and true labels:

$$\text{Jaccard} = \frac{|\{T\} \cap \{P\}|}{|\{T\} \cup \{P\}|}. \tag{16}$$

Precision is the number of true outcomes within the predicted outcomes. If no outcome is predicted and true outcomes exist, the precision is assumed to be zero:

$$\text{precision} = \frac{|\{T\} \cap \{P\}|}{|\{P\}|}. \tag{17}$$

Recall is the proportion of correctly predicted labels to the number of all true labels:

$$\text{recall} = \frac{|\{T\} \cap \{P\}|}{|\{T\}|}. \tag{18}$$

As the number of true labels differs between the different events, we first calculate the metrics for the individual events and then average over events.

3 Application

3.1 Working example

Our working example with 10 different sentences (see Table 1) comprises 771 events, 104 different cues and 20 different outcomes. On average each of the 771 events has 16.3 cues and 4.24 outcomes. The average number of outcomes (labels) is known as the label cardinality. The label density is the average number of outcomes divided by the total number of different outcomes. For the present example, the label density is 0.212.

The left part of Table 2 presents the values of the four metrics for classification. The metrics are calculated by using a cutoff value of 0.4. Performance is good according to all metrics, with classification based on $\boldsymbol{a}$ slightly outperforming the prediction vector $\boldsymbol{p}$ obtained with incremental learning. Classification is completely error-free, however, when the weight matrix $\boldsymbol{W}^{\text{end}}$ of the first network is used to calculate $\boldsymbol{A} = \boldsymbol{W}^{\text{end}}\boldsymbol{C}$. Application of equation (14) yields the estimate of $\boldsymbol{D}$, from which the prediction vector $\boldsymbol{p}'$ is obtained straightforwardly.

Table 2: Classification performance according to four metrics, for the small working example (left) and the TASA corpus (right), using activation ($\boldsymbol{a}$) or prediction values ($\boldsymbol{p}$ for incremental learning, available only for the working example, and $\boldsymbol{p}'$ using equation (14) to estimate $\boldsymbol{D}$) with a cutoff value of 0.4. For the TASA corpus, values are reported based on the last 1000 learning events and the first 1000 learning events, in this order.

metric	working example			TASA corpus	
	$\boldsymbol{a}$	$\boldsymbol{p}$	$\boldsymbol{p}'$	$\boldsymbol{a}$	$\boldsymbol{p}'$
Hamming loss	0.00065	0.05584	0.00000	0.0000482	0.0000489
				0.0000395	0.0000354
Jaccard index	0.99676	0.74017	1.00000	0.74725	0.76113
				0.78120	0.80961
Precision	0.99676	0.91616	1.00000	0.91168	0.90778
				0.91819	0.92320
Recall	1.00000	0.81321	1.00000	0.78142	0.80414
				0.82036	0.84995

Figure 1 illustrates how learning develops in the two networks for sentences 8 and 9 in Table 1, for one random order of the 771 learning events. The left panel shows the activations according to the Rescorla-Wagner network. Solid lines represent key lexomes from sentence 8 in Table 1: KICK and BUCKET for the unintended literal reading and DIE for the intended idiomatic reading. Dashed lines represent the competitors APPLE and APPLEPIE in sentence 9. The serrated patterns of the learning curves in the left and center panels reflect the learning and unlearning that unfolds as outcomes competing for the same cues are encountered. In the initial stage of learning, the lexomes DIE and APPLEPIE encoded in the utterances are not recovered, instead, the model produces the false positives KICK, BUCKET, and APPLE). By the end of learning, the proper lexomes have higher activation, but the 'literal' lexomes remain present with relatively high activations. These kinds of developmental changes are well-documented in the child language acquisition literature, see, e.g., Ramscar et al. (2013), for detailed discussion and modeling.

The center panel of Figure 1 shows the predictions generated by the second network, trained incrementally according to equation (11). Learning does not proceed as quickly as in the first network: by the end of training, KICK and BUCKET are still preferred above DIE, and APPLE receives more support than APPLEPIE. The model has learned that an apple pie is not an apple, and that kick the bucket means die. When the frequencies of the learning events are increased ten-fold, as shown in the right panel, the literal lexomes are properly suppressed, and predictions become more similar to those of the non-incremental solution using equation (14).

An important property of this approach to language comprehension is that the correct lexomes are selected without any worries about regular or irregular verbs, literal versus idiomatic expressions, finding boundaries between words, decomposing words into parts,

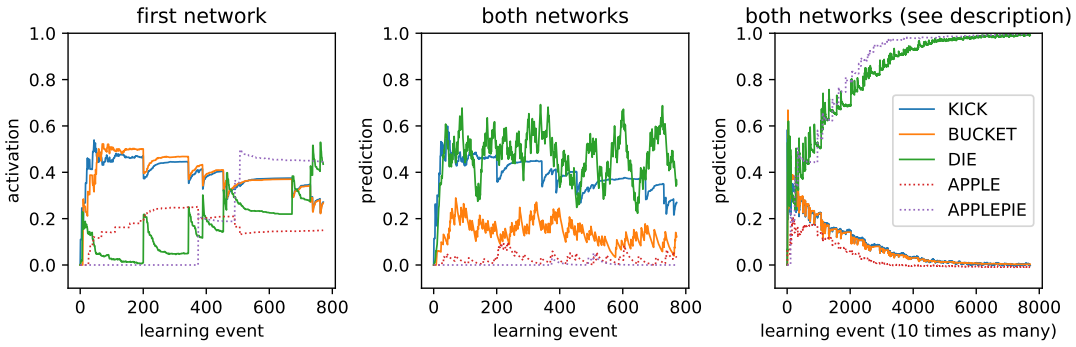

Figure 1: Activations and predictions for the selected lexomes KICK, BUCKET and DIE in the learning event of "*John kicked the bucket*" (sentence 8) and for the selected lexomes APPLE and APPLEPIE in the learning event of "*Bill ate the apple pie*" (sentence 9) in Table 1, using only the Rescorla-Wagner network (left), and the coupled networks (center and right). For the right panel, frequencies were increased tenfold to estimate the asymptotic behavior.

or disambiguating homographs. Given the assumption that understanding drives the recalibration of weights, the rich information available in the combinatorics of sublexical cues and lexomes appears sufficient for multi-label classification to be effective. This conclusion raises the question of whether this approach to language comprehension scales up to non-trivially small data sets.

3.2 A 52k multi-label classification problem

To assess whether our approach scales up to real data, we trained the model on the TASA corpus (Zeno et al., 1995), a collection of texts comprising 10,807,146 words representing 109,338 string types. Lemmatization was carried out with `TreeTagger` (`http://www.cis.uni-muenchen.de/~schmid/tools/TreeTagger/`, which distinguished 90,339 lemmata, of which 37,938 occurred once. To keep computations tractable, the model was trained on all words occurring at least twice and 351 hapax legomena that occurred in a precompiled list of words. Hapax legomena that were not included were replaced by the dummy word `HAPAX`, resulting in a total of 52,402 lexomes. Learning events were sentences in the TASA corpus. Sequences of more than 8 words were split at the next available occurrence of *and* or *or*. This resulted in a total of 993,080 learning events.

The classification problem is defined by the TASA corpus thus comprises 993,080 events, 11,725 cues, and 52,402 outcomes. Evaluation of the model trained to discriminate between all outcomes was first carried out on the last 1000 learning events. For these learning events, the average number of cues was 47.1, the label cardinality was 10.1, and the label density 0.000193. The right half of Table 2 shows good classification performance, with a low Hamming loss, precision around 0.9 and recall around 0.8. Performance based on activations and performance based on predictions, using the com-

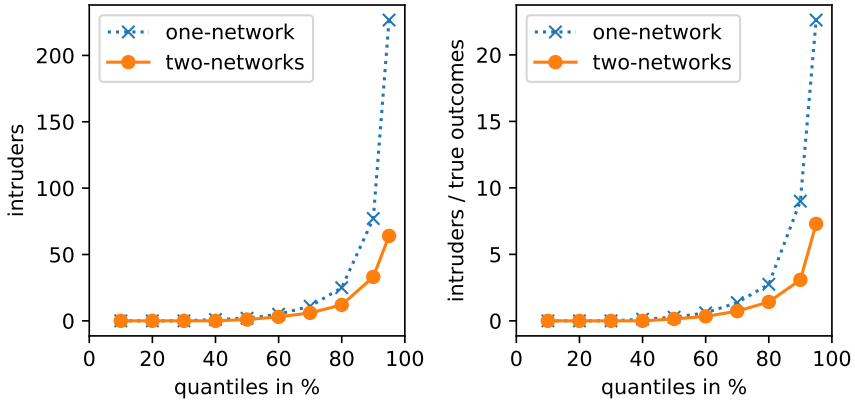

Figure 2: Deciles 0–0.9 and the 0.95 percentile for the number of intruders (left) and the ratio of intruders to true outcomes (right) for the one-network and the two-network model evaluated on the first 1000 learning events in the TASA corpus. The three largest numbers of intruders for the one-network model were 2070, 52333, and 52385. For the two-network model the three largest numbers of intruders are much smaller (328, 412, and 1275).

putational shortcut in equation (14), is very similar. The even rows of Table 2 list the same performance metrics also for the first 1000 learning events, learning events that were subsequently followed by a 992,080 further learning events. Results were nearly identical. (As calculation of the metrics for a single learning event requires about 12 seconds, and to keep the carbon footprint of this study within bounds, we did not evaluate performance for all 993,080 learning events.)

It is also informative to assess model performance from the perspective of the number of intruders, i.e., lexical candidates that have activation or prediction values that are higher than those of at least one true outcome. The left panel of Figure 2 shows that for around 40% of the learning events, there are no intruders at all. More than 100 intruders (out of 52k possible intruders) are present only for 28 learning events. The right panel presents the ratio of intruders to true outcomes. It is only for deciles > 0.8 that the number of intruders is larger than the number of true outcomes in the learning event. Examples of intruders are DOWN for the sentence "*The aleuts were housed in abandoned rundown gold mines or fish canneries*", and FIELD and SUCCESS for the sentence "*He is an ecologist who studied succession in abandoned cornfields*". The intruders are high-frequency words that are part of less-frequent complex words.

We have evaluated model performance on samples of the learning events on which the model was trained. Cross-validation is possible, but a complicating factor is that decisions have to be made concerning out-of-vocabulary words, i.e., words in out-of-bag samples that are not available to the model during learning. Such words can be discarded during evaluation; experiences with a model for auditory comprehension using only the first weight matrix suggest good generalization performance under this form of cross-

Table 3: Performance measures for 8866 words presented in isolation to the model.

	hamming	jaccard	precision	recall	median number of intruders
activations	0.000023	0.0949	0.0949	0.0959	13
predictions	0.000023	0.1825	0.3361	0.1868	7

validation (Arnold et al., 2017). A further complication that arises when evaluating model performance is that a few words are re-used time and again, while many words occur very infrequently. Since words that are encountered rarely have little chance of being learned well, the misses will tend to be low-frequency words and the false alarms orthographically similar higher-frequency words.

The performance of the model does not depend on 'memorizing' individual learning events. Instead, the model is productive in the sense that it can deal with novel utterances, provided the lexomes have been encountered during training. By way of illustration, consider the sentence "*After playing the boys and girls went home to eat*", which does not occur in the TASA corpus. There are no intruders for this sentence: The encoded lexomes have the highest activations of all 52-k lexomes. Given the 0.4 threshold used above, EAT is a miss, but its prediction value is, with 0.397, very close to the threshold. Furthermore, the next most-activated competitor is EACH, which, however, has much lower support (0.155). Thus, at least for this example, the model is highly successful in discriminating between the intended and unintended lexomes.

The combined networks perform slightly better in terms of recall and the Jaccard index. For precision and Hamming loss, results are ambiguous. This raises the question of whether the second network is actually necessary. Interestingly, the second network turns out to be a true enrichment to our model, for two reasons. The first reason is that when words are presented in isolation to the network, without the supporting cues from the other words in the context, the second network is essential.

Table 3 summarizes the performance for 8866 words presented in isolation to the model. Precision and recall are almost twice as high when based on predictions instead of activations, and the median number of intruders is reduced from 13 to 7.

For a sample of 100 words that have at least one intruder, we calculated activations and predictions as well as numbers of intruders when presented in isolation and when presented together with other words in a (randomly selected) utterances containing the targeted word. The sample was create by randomly selecting 10 words out of the groups of 1, 2, ..., through 9 intruders and by selecting 10 words out of all isolated words that had more than 10 intruders when presented in isolation. Intruders for the presentation within the utterance were only counted if they had a rank lower than the target word. Figure 3 shows the distribution of activation and prediction values (left) and the number of intruders (right). Distributions of activation values are shifted down compared to distributions of prediction values, and utterances outperform single-word presentation. This indicates that the cues of other words are co-learned with the cues of the target word, strengthening its discriminatibility. The right panel of Figure 3 shows that the number of intrusions tends to be higher for activations than for predictions, and that

there are more intruders for utterances than for single-word presentation. Given the larger number of words in utterances, this is unsurprising.

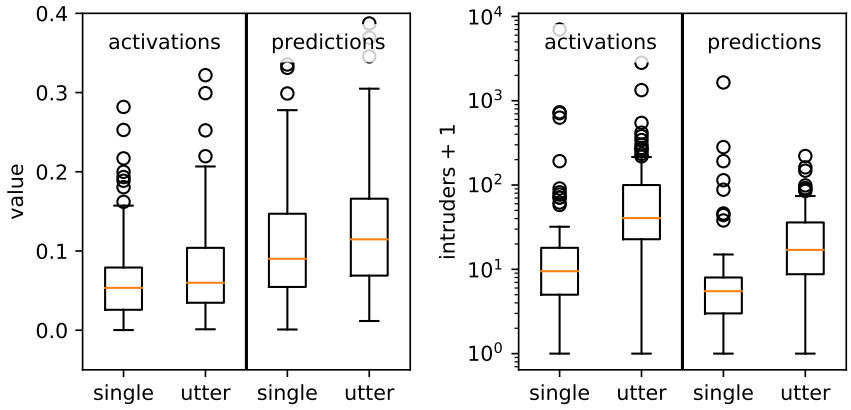

Figure 3: Values of activations c.q. predictions (left) and number of intruders+1 (right, log-scale) cross-classified by presentation as single words versus presentation in utterances. Values increase for prediction for utterances, but due to multiple words being present in utterances, the number of intruders also increases.

A second reason for maintaining the second network is that it turns out to be useful for defining high-dimensional spaces in which words that are perceived to be more similar in meaning tend to be more strongly correlated.

Given a set of k unique learning events and m unique outcomes, an $m \times k$ matrix $\boldsymbol{P}$ defines, for each event, the predictions $\boldsymbol{p}$ for all m outcomes. As the number of learning events in the TASA corpus is close to a million, it was necessary to reduce the number of learning events to keep computations tractable. We therefore randomly selected two learning events for the same 8866 words mentioned above, resulting in a total of 17,152 learning events (in 580 cases there was overlap with two or more lexomes in the same event). The total number of outcomes in this subset of learning events was 19,251. We trained the first network on the TASA data as described above, and then used the more restricted subset to calculate the matrices **A** (19,251 lexomes $\times$ 17,152 learning events), **D** (19,251 $\times$ 19,251 lexomes) and **P** (19,251 lexomes $\times$ 17,152 events).

P defines a semantic vector space (Landauer and Dumais, 1997): correlations between the row vectors of **P** predict the perceived semantic similarity between words. We illustrate this for the semantic similarity ratings collected by Bruni et al. (2014). A generalized additive model (Wood 2017, for the evaluation of significance of smooth see Wood 2012) fitted to the human similarity ratings for 2369 word pairs, using as predictor the correlations of the corresponding row vectors of **P**, yields the partial effect shown in Figure 4. For 90% of the data points, a nearly linear relation is observed, with larger positive correlations predicting higher reported semantic similarity.

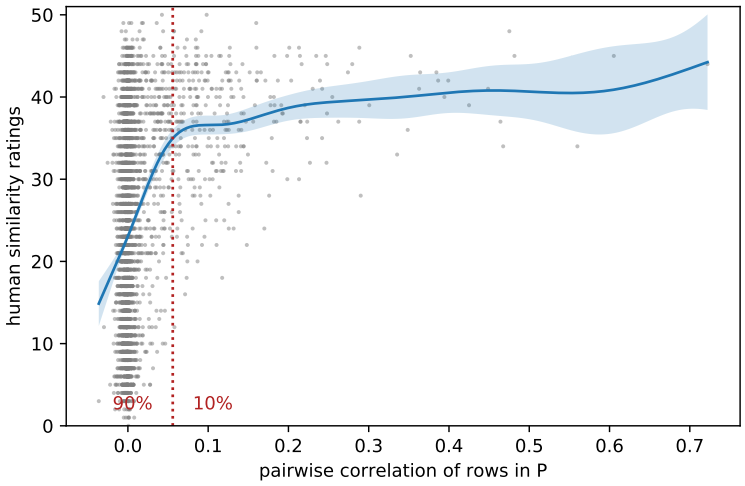

Figure 4: Partial effect in a generalized additive model (gam) of the correlations between pairs of row vectors of **P** as predictors of human semantic similarity ratings for the corresponding pairs of words. The red vertical line indicates the 90% percentile. The gam was estimated with the **mgcv** package for R modeling both the mean and variance as nonlinear functions of the correlations using the gaulss family (edf = 7.485, ref.df = 8.386, $\chi^2 = 1278.7$, $p < 0.0001$).

4 Concluding remarks

Multi-label classification is a hard problem, not only for statistics, but also for humans. For instance, in auditory word recognition, isolated words taken from conversational speech have recognition rates between 20% and 40% (Arnold et al., 2017). In the visual lexical decision task, undergraduate students perform near chance on the lower-frequency words (Baayen et al., 2017). From this perspective, the model's performance, with training on a mere 10 million words, is remarkably successful. Given that in lexical decision tasks with many items, undergraduate students classify some 20% of the words presented to them as nonwords, and given that in single-word presentation the eighth decile is at 47 intruders, we speculate that in this task smaller numbers of intruders are tolerated, but larger numbers of intruders lead to false rejections.

Given that the model presents a simplified perspective on the first stage of comprehension — understanding the words — several of its features are remarkable. First, the traditional linguistic assumption that language is a (de)compositional system is replaced by a perspective in which the language signal is a code that discriminates between possible messages (Ramscar, 2013; Shannon, 1956). Here, we have shown that sublexical features of surrounding words enhance discrimination.

Second, the model is parsimonious in its parameters. For each network, there is only one free parameter, the learning rate η. We used the same learning rate for both networks, thus, the weights **W** and **D** are determined almost completely by the data.

It is worth noting that although **W** and **D** can be very large, most of the weights are very close to zero. For instance, for **W**, only 4496 weights exceed 0.1 (0.000000073% of the total number of weights), and only 191 weights are greater than 0.5. Arnold et al. (2017) show for auditory comprehension that **W** can be pruned down to a fraction of the original weights without noticeable loss of accuracy.

Third, the classifier implements a three-layer network that differs from backpropagation networks in that there is direct error injection twice, once for **W** using the Rescorla-Wagner equations, and once for **D**, using Widrow-Hoff (or the generalized inverse). The second matrix makes classification robust when words are presented in isolation, without the sentential context in which they are normally embedded.

Fourth, more sophisticated features than letter trigrams can be used as cues, such as the frequency band summary features used by Arnold et al. (2017) for modeling auditory word recognition, and for reading the histogram of oriented gradients feature descriptor proposed by Dalal and Triggs (2005) and implemented in Linke et al. (2017) for predicting lexicality decisions in baboons.

Fifth, the model promises to scale up to realistic data sets. The 52-k classification problem addressed in this study, although not trivial, is still at the lower boundary of the lexical knowledge that speakers have at their disposal, and it remains to be shown that the present approach will work as well for 100-k multi-label classification problems. The problem will become harder, but then, there is a cost to knowing more also for speakers, as is evident of the increasing costs of the accumulation of knowledge over the lifespan (Ramscar et al., 2014).

Finally, the model offers a more dynamic perspective on the vexed question of what words' meanings actually are. Although we talk about words as if they 'have' meanings, and that these meanings are fixed and immutable as in printed dictionaries, the context in which words appear is a crucial for proper interpretation (Firth, 1968). In our approach, the sublexical cues of a full utterance give rise to a pattern of predictions over all lexomes, a pattern that we anticipate will differ depending on whether the input is auditory or visual. This distribution of predictions in turn creates a distribution over experienced events. In the present implementation, we selected a large number of event 'exemplars' that were a subset of the total number of events. It is computationally infeasible, and cognitively implausible, to work with prediction vectors with the dimensionality of all events encountered. In human memory, events cluster and merge, and we suspect that attentional mechanisms restrict the event space even further. A topic for further investigation is how to properly reduce the event space and how to allow attention to zoom in on further subsets of events. In such a system, meaning is the state of the event space that the system is moved into after experiencing the input. Importantly, in this approach, lexomes are not 'meanings' in the dictionary sense — they are theoretical constructs that are the crutches that we have to lean on to move forward towards a formalization of meaning in terms of the state of a high-dimensional system, in the hope that future research will allow us to get rid of the construct altogether.

References

D. Arnold, F. Tomaschek, F. Lopez, Tino Sering, and R. H. Baayen. Words from spontaneous conversational speech can be recognized with human-like accuracy by an error-driven learning algorithm that discriminates between meanings straight from smart acoustic features, bypassing the phoneme as recognition unit. *PLOS ONE*, 12(4):e0174623, 2017. URL `http://journals.plos.org/plosone/article?id=10.1371/journal.pone.0174623`.

R. H. Baayen, F. Tomaschek, S. Gahl, and M. Ramscar. The Ecclesiastes principle in language change. In M. Hundt, S. Mollin, and S. Pfenninger, editors, *The changing English language: Psycholinguistic perspectives*, page in press. Cambridge University Press, Cambridge, UK, 2017.

E. Bruni, N.K. Tran, and M. Baroni. Multimodal distributional semantics. *Journal of Artificial Intelligence Research*, 49:1–47, 2014.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893, 2005.

John Rupert Firth. *Selected papers of J R Firth, 1952-59*. Indiana University Press, 1968.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

T.K. Landauer and S.T. Dumais. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction and representation of knowledge. *Psychological Review*, 104(2):211–240, 1997.

M. Linke, F. Broeker, M. Ramscar, and R. H. Baayen. Are baboons learning "orthographic" representations? probably not. *PLOS-ONE*, 12(8):e0183876, 2017.

M. Ramscar. Suffixing, prefixing, and the functional order of regularities in meaningful strings. *Psihologija*, 46:377–396, 2013.

M. Ramscar, Melody Dye, and Stewart M McCauley. Error and expectation in language learning: The curious absence of mouses in adult speech. *Language*, 89(4):760–793, 2013. doi: 10.1353/lan.2013.0068.

M. Ramscar, P. Hendrix, C. Shaoul, P. Milin, and R. H. Baayen. Nonlinear dynamics of lifelong learning: the myth of cognitive decline. *Topics in Cognitive Science*, 6:5–42, 2014. doi: 10.1111/tops.12078.

R. A. Rescorla and A. R. Wagner. A theory of Pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. In A. H. Black and W. F. Prokasy, editors, *Classical conditioning II: Current research and theory*, pages 64–99. Appleton Century Crofts, New York, 1972.

Konstantin Sering, Marc Weitz, David-Elias Knstle, and Lennart Schneider. Pyndl: Naive discriminative learning in python, January 2018. URL `https://doi.org/10.5281/zenodo.1134829`.

Claude E Shannon. The bandwagon. *IRE Transactions on Information Theory*, 2(1):3, 1956.

Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. *1960 WESCON Convention Record Part IV*, pages 96–104, 1960.

S. N. Wood. *Generalized Additive Models*. Chapman & Hall/CRC, New York, 2017.

Simon N Wood. On p-values for smooth components of an extended generalized additive model. *Biometrika*, page ass048, 2012.

S.M. Zeno, S.H. Ivens, R.T. Millard, and R. Duvvuri. *The educator's word frequency guide*. Touchstone Applied Science, New York, 1995.