Psychological Methods
2016, Vol. 21, No. 4, 542–565

1082-989X/16/$12.00 http://dx.doi.org/10.1037/met0000098

Comparing Vector-Based and Bayesian Memory Models Using Large-Scale Datasets: User-Generated Hashtag and Tag Prediction on Twitter and Stack Overflow

Clayton Stanley and Michael D. Byrne
Rice University

The growth of social media and user-created content on online sites provides unique opportunities to study models of human declarative memory. By framing the task of choosing a hashtag for a tweet and tagging a post on Stack Overflow as a declarative memory retrieval problem, 2 cognitively plausible declarative memory models were applied to millions of posts and tweets and evaluated on how accurately they predict a user's chosen tags. An ACT-R based Bayesian model and a random permutation vector-based model were tested on the large data sets. The results show that past user behavior of tag use is a strong predictor of future behavior. Furthermore, past behavior was successfully incorporated into the random permutation model that previously used only context. Also, ACT-R's attentional weight term was linked to an entropy-weighting natural language processing method used to attenuate high-frequency words (e.g., articles and prepositions). Word order was not found to be a strong predictor of tag use, and the random permutation model performed comparably to the Bayesian model without including word order. This shows that the strength of the random permutation model is not in the ability to represent word order, but rather in the way in which context information is successfully compressed. The results of the large-scale exploration show how the architecture of the 2 memory models can be modified to significantly improve accuracy, and may suggest task-independent general modifications that can help improve model fit to human data in a much wider range of domains.

Keywords: ACT-R declarative memory theory, vector-based models, LSA, machine learning

With the massive growth of human-created content on social media sites (e.g., Twitter, Stack Overflow, Wikipedia) and public access to many of those databases, it is now possible to evaluate psychological theories of declarative memory on large-scale real-world tasks where the theory can be tested on hundreds of millions to billions of data points. Without using large data sets it is difficult to verify and explore modifications to the declarative memory architecture because the data sets are not large enough or rich enough to rigorously test modifications to the theory.

A few researchers have started to use large-scale databases to evaluate how well declarative memory models scale, for both ACT-R Bayesian models (Douglass & Myers, 2010; Fu & Pirolli, 2007; Pirolli & Fu, 2003; Stanley & Byrne, 2013) and vector-based memory models (Jones & Mewhort, 2007; Rutledge-Taylor, & West, 2008; Recchia, Jones, Sahlgren, & Kanerva, 2010; Sahlgren, Holst, & Kanerva, 2008).

This study expands on previous work that has tested retrieval models on large-scale data sets. It focuses specifically on how two state-of-the-art declarative memory retrieval models (ACT-R Bayesian and random permutation vector-based) can be improved by testing these models on two real-world tagging tasks. One primary objective is to compare the random permutation vector-based model with ACT-R's Bayesian declarative memory theory when large data sets are used for each model. Another objective is to improve each model, and measure how incorporating the strengths of each model into the other (i.e., modifying model components) influences accuracy. For example, modifying a vector-based model to retrieve results that are based not only on context, but also on the prior odds that a particular item in memory will be needed again. Large-scale data sets were used throughout the model development and evaluation process. This provides an ideal testing environment for constraints and assumptions for each theory to be rapidly tested and new model components to be hypothesized and quickly evaluated.

As a general methodology for this research, we used a cognitively constrained exploratory model development process. Because we worked with large-scale data sets, we had enough data to explore many different modifications simultaneously without the risk of overfitting. However, we were not interested in bringing to bear the entire collection of machine learning algorithms to this prediction task in order to completely optimize a model. Rather, we were interested in evaluating how well two cognitive models could explain the results, and then making selective modifications to them to see how performance changes. The modifications that we explored were selected such that they influenced model accuracy, could be easily added to (or removed from) the cognitive

Clayton Stanley and Michael D. Byrne, Department of Psychology, Rice University.

Correspondence concerning this article should be addressed to Clayton Stanley, Department of Psychology, Rice University, 6100 Main Street, Houston, TX 77005. E-mail: cstanley@cstanley.no-ip.biz

models without massive modifications, and could be considered cognitively plausible.

Research Questions

The core question motivating this research is: What kinds of cognitively plausible models can best predict the tags that a user on a social media site selects? This question can be tested by generating a prediction for the most likely selected tag whenever a user is about to generate a tag, and then comparing the user's chosen tag with the model's prediction. The process of suggesting a relevant tag to a user can be thought of as a memory retrieval request for a tag, given prior tag use and current context.

Co-occurrence-based modeling has been shown as a potentially useful approach when predicting Twitter hashtag selection (Efron, 2010) and Stack Overflow tag selection (Stanley & Byrne, 2013). Several memory retrieval models are based on this broad methodology, where a count is maintained of the times each contextual element (such as a word in a post) co-occur with a tag. At least two of the current memory retrieval models are based on a co-occurrence methodology: ACT-R's declarative memory (DM) retrieval theory and random permutation vector-based memory systems.

These two memory models were compared across two domains where users generate tags for posts: Twitter and Stack Overflow. Twitter is a microblogging service where users create 140 character "tweets" and broadcast those messages out to the users who follow their posts. A user is free to annotate important concepts or words in the tweet by preceding words with "#," thereby making the word into a hashtag. After the tweet is created the hashtags become hyperlinks, and if clicked on will take a user to a summary feed of all of the tweets across Twitter that have used that hashtag. Some recent and often used hashtags for Twitter are *#photography*, *#startup*, *#4change*, *#android*, and *#solar*.

The two memory retrieval models were also tested on tag selection for Stack Overflow posts. Stack Overflow is a question and answer site for computer programming where users ask programming-related questions and fellow members of the community provide answers. A user with a programming-related question creates a post with their question and tags the post with a few specific programming-related tags. The tags chosen by a user on the Stack Overflow site represent the primary topic of the question, such as a specific programming language, tool, software package, or framework. Some example tags for Stack Overflow are *PHP*, *Arrays*, *MVC*, *C#*, and *Common Lisp*.

The Stack Overflow and Twitter data sets were chosen because the human-created content between them is quite different, and were interested in retrieval models that generalize across tasks. However, they are also similar on several accounts: Both domains have amassed large amounts of user-created content, users generate tags for posts when creating content on the site (hashtags for Twitter and tags for Stack Overflow), and the user data from the data sets are publicly available for analysis.

Roadmap

This rest of this document is separated into five sections. The Theoretical Background section describes the vector-based and Bayesian declarative memory models that were evaluated in this research, shows how these models are related to the other common declarative memory models, and identifies examples of similar tagging models that have already been applied to various tag recommendation systems. The General Methodology section presents the overall method for testing the models on the Stack Overflow and Twitter data sets. The Past User Behavior section shows the results for the first test of the models, where the model term for past user behavior is examined in isolation, and tags are predicted based purely on a user's past tagging history. The Combining Predictors section adds the context component to the models, and shows how model accuracy is influenced by changing various architectural components of each model. The final section provides a general discussion of the overall findings.

Theoretical Background

ACT-R Declarative Memory Theory

ACT-R (Anderson, 2007) is a cognitive architecture that formalizes how each cognitive process of the mind (e.g., memory, learning, visual, and motor) interacts to produce behavior. The declarative memory system is a component of the architecture that models the timing, learning, and forgetting processes that make up declarative memory storage and retrieval. At the core of the system is the concept of a "chunk:" the representation of a piece of knowledge (e.g., a word). The equations that make up this system were derived from a rational analysis of declarative memory retrieval (Anderson & Milson, 1989). That is, given the task of retrieving a chunk of information from declarative memory, the current context (i.e., external and internal environment state), and past experience (i.e., prior memories and exposure), what is the optimal behavior (i.e., the optimal chunk to retrieve from memory)? Using Bayesian reasoning, each chunk of information in declarative memory can be assigned a prior likelihood of needing retrieval again, given the prior history of exposure to the chunk. These chunk prior probabilities are then adjusted for the current context, so that the posterior probabilities represent the likelihood that a chunk is needed, given prior odds and adjusted by current environment state.

ACT-R DM model. A formal description of the ACT-R Declarative Memory model is included in Table 1

Table 1
ACT-R Declarative Memory Model

Common name	Equation
Activation	$A_i = B_i + \sum_{j \in c} W_j S_{ji}$
Attentional weight	$W_j = \frac{W}{n}$
Base level	$B_i = log \sum_{j=1}^{n} t_j^{-d}$
Strength of association	$S_{ji} = log\frac{p(i \mid j)}{p(i \mid \bar{j})} \approx log\frac{p(i \mid j)}{p(i)} = log\frac{NN_{ji}}{N_{Row}(j)N_{Col}(i)}$
Recall probability	$P_i = \left(1 + e^{\frac{\tau - A_i}{s}}\right)^{-1}$

The total activation (A_i) for a chunk in declarative memory is a function of two components: base-level activation (B_i) and strength of association (S_{ji}). The recall probability (P_i) that a chunk will be retrieved from memory increases with total activation (A_i). Each term in Table 1 will be discussed in greater detail to follow.

Base-level activation. Base-level activation reflects the log prior odds of needing an observed chunk again. The default way to calculate these log prior odds is to use the standard base-level equation in Table 1. This equation formalizes how log prior odds are a function of both frequency and recency of prior exposure to a particular chunk. Chunks used more frequently (either through exposure or from a retrieval) are more likely to be needed for retrieval again. However, as time progresses and a particular chunk is no longer used, the activation for that chunk decays. In this way the standard base-level equation formalizes the time dynamics of the retrieval system, where a chunk's base-level activation evolves over time, depending on its frequency and recency of use.

Strength of association. Strength of association (S_{ji}) reflects the amount of log-odds adjustment to the activation of a chunk, given the current context (i.e., external environment and internal state). Context for the Stack Overflow domain for example is represented as each word in the title and body of a post. Context for the Twitter domain are the words in a tweet. Association strength between a chunk in memory and a single contextual element can be computed directly by calculating its context-adjusted odds ratio: The likelihood a chunk occurred with the current context ($p(i \mid j)$) over the likelihood that the chunk occurred in any of the other contexts ($p(i \mid \bar{j})$). For large data sets, the likelihood that a chunk occurs in any particular context reaches near-zero values ($p(i) \Rightarrow 0$). So it can be assumed that the likelihood that a chunk occurs in any context but one ($p(i \mid \bar{j})$) is equivalent to the likelihood that a chunk occurs in any context ($p(i)$). This assumption was described when deriving the S_{ji} equation (Anderson & Milson, 1989), and used in recent research working with large-scale data sets (Douglass & Myers, 2010; Farahat, Pirolli, & Markova, 2004; Stanley & Byrne, 2013).

Connection to pointwise mutual information. Pointwise mutual information (PMI) is another co-occurrence index that measures strength of association between two terms (Farahat, Pirolli, & Markova, 2004). It is based on the co-occurrence count between pairwise observations of terms, similar to ACT-R's strength of association. The index has been used to accurately and efficiently measure strength of association information for large-scale data sets (Budiu, Royer, & Pirolli, 2007; Farahat, Pirolli, & Markova, 2004). The PMI index is presented as Equation (1).

$$PMI(y, x) = log\frac{p(y \mid x)}{p(y)} \quad (1)$$

Further, this PMI equation is identical to ACT-R's strength of association (S_{ji}) equation for large data sets. Once the number of occurrences is large enough such that the probability of observing any particular contextual element ($p(x)$) is near zero, then ACT-R's strength of association index simplifies to the PMI equation.

Attentional weight. The attentional weight term allows for some contextual components to be weighted more than others when the total S_{ji} activation is computed. This reflects the fact that the declarative memory system has limited attentional resources (W) that must be spread across all of the items in context. Most ACT-R models simply weight each contextual element equally (e.g., all words in a Stack Overflow post have equal weight), so each word's attentional weight is simply normalized by the number of items in context.

Vector-Based Memory Systems

"Holographic" memory systems (Plate, 1995) or vector-based memory systems are an alternative representation of declarative memory. They have been successfully used to predict memory retrievals that leverage word order information (Jones & Mewhort, 2007). They also have similar structure to memory models that use Latent Semantic Analysis (LSA; Landauer & Dumais, 1997).

Vector-based models represent a concept (i.e., a chunk) as a vector. The representation of the concept is distributed across all elements of the vector. This is somewhat similar to taking a column (i.e., a tag) on a word co-occurrence matrix and viewing the distribution of co-occurrence counts across the rows in the column as the representation of that concept.

For vector-based systems, each word is represented by an environment vector e_i that is nearly orthogonal to all other words' environment vectors. The number of dimensions for these vectors is much less than the number of rows in a full word co-occurrence matrix (typically around 2,000–4,000), which is how vector-based systems represent information in a much more compact space. Paired with each environment vector (e_i) is a memory vector (m_i) that contains the summed representation of all other environment vectors that have co-occurred with that e_i environment vector. These memory vectors can contain environment vectors from position-independent word co-occurrence information (c_i) and word order information (o_i), which can be combined into a single representation (m_i). Over time, memory vectors accumulate a distributed representation of the most common environment vectors that co-occurred with them.

Retrieval process. Retrieving the most likely chunk given context can be done in two ways: decoding and resonance (Jones & Mewhort, 2007). Decoding takes the memory vector for context and decodes it back into an environment vector. The cosine between that decoded environment vector and all other environment vectors is computed and ranked, and the chunk with the environment vector with the highest cosine is retrieved.

Resonance is the opposite retrieval process, where a memory vector is created from the context, and then that context memory vector is compared to all memory vectors. For example, assume that a user has written the word "zend" in a post, and she wants to assign a tag for the post. "zend" has co-occurred with the tag *PHP* many times previously, so the unordered memory vector for "PHP" (c_{PHP}) contains the unordered environment vector for *zend* (e_{zend}). To determine the most correlated memory vector with context, a memory vector is created from the current context (e_{zend}). The cosine between that context memory vector and the memory vector for *PHP* will be high, because the *PHP* memory vector (c_{PHP}) often contains the context environment vector (e_{zend}). Because the cosine is high, the tag *PHP* will most likely be returned as the vector that resonates highest with the current context.

The resonance and decoding processes are inversions of each other. Usually both resonance and decoding will return similar rank orderings of most likely chunks given context.

Word order for vector-based systems. One of the strengths of vector-based systems is that word order can naturally be represented alongside word co-occurrence in these distributed representations. With vector-based systems, order information is included by creating an environment vector ($e_{bind(i,\Phi)}$) that is a function of environment vectors for both the word (e_i) and location (Φ). This function has the useful property that the resulting environment vector and the location vector can be inverted to return the original word's environment vector. Circular convolution and inverse circular convolution (i.e., circular correlation) are used by Plate (1995) and Jones and Mewhort (2007)'s BEAGLE model to implement these operations. This allows for retrieval requests such as "playing #Φ," "just read #Φ," and "vector-based Φ systems."

Random permutation model. In order to represent word order information, a function must be used that converts an environment vector for a set of words and positions (e.g., "stack Φ") into a new uncorrelated environment vector. Sahlgren, Holst, and Kanerva (2008) used a simple random permutation method for this operation.

A formal description of the random permutation model in Sahlgren et al. (2008) is included in Table 2. With random permutations, each word environment vector (e_i) is a large sparse vector of zeros with a few one and negative one values in random locations. In order to create a new environment vector for a word that preceded another word in a sentence, that word's environment vector is shifted to the left one position. This produces a new environment vector ($e_{i^{-1}}$) for the combination of the word and the position that is uncorrelated with the original word's environment vector and all other environment vectors.

Connection to LSA. Both vector-based models and LSA calculate word similarity on a reduced co-occurrence matrix. For LSA, the original word by document matrix is compressed to a factor by document matrix and the number of factors is much less than the number of words. LSA uses singular value decomposition (SVD) to find words that have highly correlated document occurrence distributions (i.e., latent concepts), and then pools those co-occurrence counts into a single dimension. For vector-based models, each tag (document) is represented by a compressed vector with a length much less than the number of words. A vector-based approach like random permutations essentially randomizes the mapping of the words on the reduced set of dimensions, and does not take into account word similarity when generating the mapping. Nonetheless, vector-based models have been shown to perform similar to if not better than LSA (Jones & Mewhort, 2007; Sahlgren et al., 2008).

Table 2
Random Permutation Model

Common name	Equation
Activation	$A_i = r(m_C, m_i)$
Memory vector	$m_i = \sum_{i \in allpast} c_i + \sum_{i \in allpast} \sum_{l \in locations} o_{i,l}$
Unordered context	$c_i = e_i$
Ordered context	$o_{i,l} = e_{i^{-l}}$
Context memory vector	$m_C = \sum_{i \in C} c_i + \sum_{i \in C} \sum_{l \in locations} o_{i,l}$
Environment vector	$e_i = rand$

Comparison of Vector-Based Models and ACT-R

Vector-based models can be thought of as a compressed representation of the full co-occurrence matrix used for the ACT-R declarative memory system. Each model computes an activation for each word to retrieve, given context and (for ACT-R's Bayesian model) prior knowledge. Because both models produce activations for each chunk, one can be swapped out for another and used as the declarative memory component of the ACT-R infrastructure. For example, Rutledge-Taylor and West (2007) used a variant of the BEAGLE vector-based model as the DM component of ACT-R. Rutledge-Taylor and West showed that an ACT-R theory with a vector-based memory system can successfully model the fan effect (Anderson, 1974).

Base-level activation. Vector-based models and ACT-R's Bayesian DM model are not identical however, and these differences need to be addressed in order to better evaluate the relative benefits of the two models. In particular, the frequency and recency of retrievals for each chunk are not currently used for vector-based models when calculating activation.

The frequency and recency of how content has been used in the past can be a strong predictor of the future importance of that information. The recent content in tweets on a news site like Twitter should be more relevant, for example. Efron and Golovchinsky (2011) tested this claim by evaluating a set of query likelihood models on returning relevant Twitter tweets from search queries. They showed that a model incorporating recency information of the tweet produced more relevant results than models that did not pay attention to this information.

ACT-R's Bayesian model has a strong theory with respect to recency and frequency information. It uses this information to calculate a chunk's prior activation (i.e., prior likelihood of being needed again, independent of current context). Vector-based models compute activation based entirely on context, so chunks that resonate with context will be retrieved, irrespective to the frequency and recency of use of that chunk. ACT-R's prior activation can also be computed for each user, so that hashtags that have been used frequently and recently for a particular user are rated with higher activation. Vector-based models have focused on computing the associative strength of a chunk and specific contextual elements. It is unexplored how accuracy changes for these models once the information about a user's prior behavior (e.g., ACT-R's base-level activation component) is added.

Word order. One strength of a vector-based approach over ACT-R's Bayesian DM model is that word order can be naturally represented alongside unordered co-occurrence information. With the ACT-R model, for example, it is unclear how to represent that a particular word appeared just before a hashtag. To represent word order in a Bayesian model, an additional dimension could be added to the co-occurrence matrix that represents word position. However, adding word-order information to ACT-R's default Bayesian declarative memory model is not common. Instead, when word order information is used to model reading for example, production rules are defined so that words are read from left to right (e.g., Lewis, Vasishth, & Van Dyke, 2006).

Activation calculation. Vector-based retrieval models calculate chunk activation by correlating a representation of the current context with each memory item. Bayesian retrieval models like ACT-R calculate activation by computing each chunk's posterior

odd adjustment likelihood, given the context. These two computations are certainly not equivalent, even though they are most likely correlated with one another.

Using a correlation can be problematic due to range effects in observed co-occurrences. If there are particular words for a tag that co-occur much more often than all of the other words with that tag, that single large value diminishes the power of all other words to contribute to the correlation calculation. This does not happen with Bayesian retrieval models because each pair of word and tag values is normalized by the total number of words that appeared with that tag and total number of tags that appeared with that word (see the strength of association equation in Table 1). The high-frequency words are often removed from the co-occurrence matrix for vector-based models to reduce this effect.

Hashtag Prediction

Models that attempt to predict a user's chosen hashtag have recently been created and tested for a few social media sites, though most of these models are not based on theories of human declarative memory. Google has even deployed a hashtag recommendation model to help users properly tag posts on their Google + social media site (Google, 2013). A tag-suggestion tool has also been deployed for the tex.stackexchange.org and superuser.com StackExchange sites. Models for Twitter and Stack Overflow have been proposed and tested, but none have yet been deployed. Recommendation models for Stack Overflow and Twitter will be described in more detail to follow.

Stack Overflow. Stanley and Byrne (2013) used a declarative memory model to predict a user's chosen tags for a post on the Stack Overflow site. They modified the standard ACT-R model in order to more accurately predict tag use on Stack Overflow. A formal description of the modified ACT-R Bayesian model is included in Table 3.

The activation of each tag (A_i) is a function of four components: That tag's base-level activation (B_i), contextual activation for the words in the title, activation for words in the body, and an offset term (O). This model has been modified from the standard ACT-R model in four ways: Two separate contextual components were used (words in title and body of post), an offset term was added, an entropy measure was used to compute the attentional weight (W_j), and the base-level activation of a tag was based on global past tag use and not customized to a user's past tagging history.

The offset term was added to normalize the mean activation across retrievals. This allowed the use of a logistic regression statistical technique to calibrate the weights for each of the terms. The scaled entropy measure E_j was chosen for the attentional weight (W_j) term because it has a strong theoretical foundation (Dumais, 1991) and is parameter-free.

Table 3
Stanley and Byrne's Stack Overflow Tag Prediction Model

Common name	Equation
Activation	$A_i = B_i + \sum_{j \in T} W_j S_{ji} + \sum_{j \in B} W_j S_{ji} - O$
Base level	$B_i = log\frac{p_i}{1-p_i}$
Strength assoc.	$S_{ji} = log\frac{p(i \mid j)}{p(i))} = log\frac{NN_{ji}}{N_{Row}(j)N_{Col}(i)}$
Attentional weight	$W_j = W\frac{E_j}{\sum E_j}$
Scaled entropy	$E_j = 1 - \frac{H_j}{H_{max}}$
Entropy	$H_j = -\sum_{i=1}^{N} p(i \mid j) log(p(i \mid j))$

Twitter. Several hashtag recommendation models for Twitter have been developed recently. One of the first models used a Bayesian co-occurrence statistical technique to predict the most likely hashtag associated with a tweet as a function of prior hashtag use and context (Mazzia & Juett, 2009). This is quite similar to an ACT-R declarative retrieval model. The main potential difference is that—like Stanley and Byrne (2013)—the global prior likelihood of a hashtag is computed without taking into account the specific user's past history.

Several other models have taken a tweet-centered approach to hashtag prediction, where suggested hashtags are collected from hashtags used in similar tweets. Li, Wu, and Zhang (2011); Zangerle, Gassler, and Specht (2011); Kywe, Hoang, Lim, and Zhu (2012) all store a content vector for each tweet, and then compute a word co-occurrence-based similarity score between a composed tweet and the rest of the tweets in the database.

Other successful models have shown that higher-level processes such as a user's goals for picking a hashtag, demographics, and specifics about how a hashtag has been used in the past can also influence hashtag use. Lin, Margolin, Keegan, Baronchelli, and Lazer (2013) showed that there are at least two distinct use cases for hashtags on Twitter: to mark the content (traditional tagging) and to contribute to a specific community that has created a hashtag. Yang, Sun, Zhang, and Mei (2012) found that demographics such as gender, age, and location can influence a user's selected hashtag. Denton, Weston, Paluri, Bourdev, and Fergus (2015) found that the growth and lifetime of a hashtag can be influenced by factors such as how often the hashtag was retweeted and the number of followers for a user that used a particular hashtag. These types of goal-related features, demographic information, and higher-level attributes of previous hashtag use were not directly included in the declarative memory models that were tested for this research. They were not included because it would be difficult to fit these features into a modeling framework that is constrained to use only declarative memory processes, and this research focused on measuring differences in model accuracy for two declarative memory models. Nonetheless, many of these higher-level features may indirectly influence a declarative memory model by influencing the specific tags that a user has chosen in the past, which in turn boosts base-level activation for these previously chosen tags, which in turn makes it more likely that they are chosen in the future.

There has been a mix of co-occurrence models created to predict hashtag use on Twitter, and the results are encouraging. However, there has been little research on how a specific user's past hashtag use should influence the model's prediction when that user is composing a tweet. Kywe, Hoang, Lim, and Zhu (2012) did take a user-centered view with their model, but the hashtags predicted by the user and content were analyzed separately and then the top from each group were combined for prediction. The ACT-R declarative memory retrieval theory shows how past experience and

current context combine to produce the most likely retrieval. Each component is summed together to generate a total activation, and then the likelihood of chunks are ranked by that summed activation. This technique is compensatory and allows for a more natural combination of the two components, where weights can be assigned to each component depending on how strongly each term predicts performance.

General Methodology

Different subsets of the Stack Overflow and Twitter data sets were used to evaluate past user behavior in isolation and to test the context component of each memory model. These data sets will be named and described in general terms. Also, the general methods used to tokenize and stem the words in all of the data sets will be described. All software code created for this research is publicly available on GitHub (Stanley, 2014).

Models

Two models were evaluated for hashtag prediction on both the Twitter and Stack Overflow data sets: the random permutation vector-based model (Sahlgren et al., 2008) in Table 2 and the modified ACT-R Bayesian model (Stanley & Byrne, 2013) in Table 3.

Data Sets

The most recent quarterly published Stack Overflow dataset (StackExchange, 2014) was used to test the models on user chosen tags for posts. This dataset contains around 5.7 million posts and 34,377 unique tags.

Unlike Stack Overflow, the entire Twitter dataset is not released to the public. Due to this and the scale of Twitter, only subsets of the Twitter dataset were examined. The relevant subsets are described below.

Popular users dataset. A popular-users dataset was collected for both Stack Overflow and Twitter. This dataset consists of all of the tweets and posts from various samples of the top users for each site. This dataset was collected by fetching all past tagging history for each of the top users. It was used to explore the growth and decay characteristics of a hashtag for individual users. The temporal dynamics of ACT-R's decay rate model for a chunk were evaluated against these data. Multiple samples were taken across a broad range of user types for each dataset to ensure that the model results were not dependent on specific attributes in a single sample.

For Stack Overflow, top users were identified by their reputation score on the site (based on their upvoted questions and answers) and the total number of questions asked on the site. To create the Stack Overflow popular-users dataset, a total of 12 slices of users were taken at different levels of reputation and total number of questions asked. Five-hundred users were sampled that had a reputation higher than 500, 1,000, 5,000, 10,000, 50,000, and 100,000. One-hundred users were sampled that had asked a total number of questions higher than 50, 100, 200, 300, 400, and 500.

For Twitter, top users were those that had the largest number of followers and had authored the largest number of tweets. To create the Twitter popular-users dataset, 12 slices were also taken, this time at different levels of number of followers and total number of tweets. One-hundred users were sampled that had a total number of followers higher than 1,000, 5,000, 10,000, 100,000, 1,000,000, and 10,000,000, and a total number of tweets higher than 100, 500, 1,000, 5,000, 10,000, and 50,000.

Popular hashtags dataset. An additional popular-hashtags dataset was collected for Twitter. This dataset was used to test the context component of the models for Twitter. The dataset contains approximately three million tweets where at least one of the 400 most popular hashtags are used. A quota of three million tweets was selected because previously research testing Bayesian co-occurrence models on large-scale data sets has used on the order of one million documents (Budiu et al., 2007; Douglass & Myers, 2010; Stanley & Byrne, 2013). Tweets were collected in real time until the tweet quota was reached.

Four popular-hashtags data sets were collected at four different time periods, spanning 1 month. Multiple data sets were collected to ensure that the results were not dependent on a particular sample of hashtags and would generalize across different popular-hashtag data sets and different time periods. Each dataset required around four days to collect enough tweets to reach the quota, and one collection was done each week.

Randomly sampled dataset. Because the entire dataset is available for Stack Overflow, randomly sampled posts were used to test the context models rather than creating a popular-tags dataset (analogous to Twitter). If all Twitter data were available, then the models would have been tested on a randomly sampled dataset for Twitter as well. However, because this is not the case for Twitter, the popular-hashtags dataset was used instead of a randomly sampled dataset for that site, while models for Stack Overflow were tested on the more challenging dataset where posts were randomly sampled across all posts created on the site.

Acquiring Data Sets

Stack Overflow. The entire Stack Overflow dataset is packaged and released quarterly to the public, so the most recent packaged dataset was downloaded and the tables were imported into a relational database.

Twitter. Twitter provides two types of APIs for acquiring data: one based on streaming and a direct query-based interface. The query-based interface was used to collect all tweets for specific popular users. At most 3,000 of the most recent tweets were collected for each user, because Twitter only allows access to this number of tweets for each user.

The streaming interface was used to collect a sample of all of the tweets that included one of the top 400 hashtags at that time. Twitter's streaming API only allows at most 400 hashtags to be monitored at a time. Otherwise a larger number of hashtags would have been collected. Twitter limits the streaming interface to randomly sample about 1% of all of the real-time data that is generated on Twitter. This sampling rate was adequate to collect around three million tweets within a few days.

Tokenization

A custom tokenization algorithm created by Owoputi et al. (2013) was used to parse the words in the text. This algorithm was specifically created for tokenizing Twitter content. However, during testing it was also found to be robust and well-suited for Stack

Overflow content. The initial model results with the new tokenizer were qualitatively similar to the model results found in Stanley and Byrne (2013) where the Python Natural Language Processing (NLP) toolkit (Bird, Klein, & Loper, 2009) was used. The same tokenization algorithm from Owoputi et al. (2013) was used for both Stack Overflow and Twitter to make comparisons across data sets more equitable.

Stemming

Owoputi et al. (2013)'s algorithm handles basic word stemming (e.g., removing "ed" and "ing" endings). With the Stack Overflow dataset, however, it is also possible to convert synonym tags to canonical base tags, since the community maintains a tag-synonym database. This can be thought of as semantic stemming, where two words with identical meaning are converted to the same canonical word. Therefore tags used for the post were converted to their root tag when possible. This does make the decision process easier for the Stack Overflow dataset. However, we chose to include this to help ensure that model predictions would stabilize. This provided the best opportunity to reliably observe differences in accuracy when altering the models.

Testing for Reliable Differences

In similar spirit to Masson and Loftus (2003), Tryon (2001), and Cumming and Finch (2005), confidence intervals were used throughout for statistical inference instead of explicit p values. This was done because it is a more straightforward way to visualize all of the mean differences in accuracy (i.e., bar and line plots with a visual indication of the error) for each of the models, which eases interpretation. It is also a conservative estimate of reliability, as using confidence intervals on the means essentially ignores the fact that this is a within-subjects design, where accuracy for each model is measured for each subject in each dataset. However, we would rather err on the side of making the results easier to interpret at the cost of using conservative estimates for reliability (i.e., use error bars overlaid on a visualization of the means instead of statistical tests on every combination of difference scores). In fact, using large data sets for analysis is what empowers us to do this: With this much data, we can use straightforward (although more conservative) inferentials and still see reliable results. This allows us to concentrate more on proper data visualization of the main findings.

Nonetheless, when accuracy differences between two models are called out explicitly in the text, the confidence interval for the mean difference in accuracy is also included. Because this interval is drawn across the observed difference scores, it will be smaller than the intervals included in the plots, since these are drawn independently for each model's overall accuracy.

Past User Behavior

How does a user's prior tag use influence their likelihood of choosing that tag in the future? This question was tested by examining how well ACT-R's base-level activation component of declarative memory predicts the tags that authors choose, given each author's past tagging behavior. The model predicts that the more recently and frequently used tags have higher prior activations, and are more likely to be needed in the future.

Table 4
Base-Level Component of ACT-R Declarative Memory

Common name	Equation
Base-level activation	$B_i = log\sum_{j=1}^{n} t_j^{-d}$
Optimized learning	$B_i = log\frac{n}{1-d} - d * logL$
Hybrid	$B_i = log\left[\sum_{j=1}^{k} t_j^{-d} + \frac{(n-k)(t_n^{1-d}-t_k^{1-d})}{(1-d)(t_n-t_k)}\right]$

Base-Level ACT-R Model

The base-level activation component from the full ACT-R declarative memory model is included in Table 4.

All three terms are methods for computing the base-level activation (B_i) of a chunk in declarative memory. The standard form has a time dependence and accounts for the fact that the rate of presentations for a given object (e.g., the word "loop") can change over time. The optimized learning form assumes that this rate is constant. The hybrid form is a blend of the standard and optimized form where activation for the most recent k observations are computed using the standard form and the rest use the optimized form.

For all equations, i represents each chunk (e.g., the word "loop"), j is a specific observation for a chunk (e.g., the second time that "loop" appears), and n is the number of times each chunk has been observed. Because the optimized learning form assumes a constant presentation rate for each chunk, it only has to keep track of the time since the first presentation of a chunk (L) and the total number of instances observed for that chunk (n). It does not have to keep track of the time each specific chunk instance was observed. Consequently, the optimized learning form of B_i is computationally much less expensive than the standard form. The hybrid form only has to keep track of the most recent k presentations of each chunk, so it only pays a small computational penalty over the optimized learning form. The optimized learning form is set to the default in the ACT-R Common Lisp implementation.

The decay rate parameter (d) represents how fast the activation for an individual presentation of a chunk decays from declarative memory. Higher values mean that chunk activation decays faster, which means that the most recently presented chunks will have the highest activation, and the total number of times that a chunk has been presented matters less. Lower decay rate values flip the importance of frequency and recency.

The base-level (B_i) component in Table 4 is different than the B_i used in Stanley and Byrne (2013) (see Table 3). Stanley and Byrne assumed no time dependence, no decay rate, and that the B_i is simply a log-odds computation of the probability that each chunk is likely to be presented again, given past frequency. If a decay rate of 0 is used on the optimized learning form of B_i, then the equation collapses to $log(n_i)$, which is a nonlinear transformation of the B_i used by Stanley and Byrne ($log\frac{p_i}{1-p_i}$). Both forms maintain the same rank ordering, so they should produce similar results. The B_i component used by Stanley and Byrne can be roughly thought of as the standard form of B_i when the decay rate is 0 (i.e., a pure frequency model).

Method

The Stack Overflow and Twitter popular-users data sets were used to characterize tag growth and decay for a user on each site. Each user's tag behavior from each site was analyzed.

Twenty-two different values of the decay rate parameter were explored (0, .1, .2, .3, .4, .5, .6, .7, .8, .9, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.8, 2, 5, 10, 20). The values were chosen to look at how performance changes from a pure frequency model ($d = 0$), to a blend of frequency and recency ($d \approx .5$), to a pure recency model ($d = 20$). The decay rate value was varied for the standard form, the optimized learning form, and the hybrid form of B_i. Only values below 1 were examined for the optimized and hybrid forms, since values above 1 for these forms are not mathematically defined. They are not mathematically defined at a decay rate greater than 1 because for these values the (1–d) denominator flips the ratios to negative values (see Table 4). At a decay rate equal to 1, every base-level activation B_i is infinite, which is not psychologically interesting.

All three forms of B_i were used to predict each user's specific tag history. To do this, a user's entire tag history was collected, and then each tag instance from oldest to newest was examined. For each of these tagging instances, the model generated an activation value for each possible tag based on that tag's frequency and recency of past use.

At each tagging instance, the model has activation values for a set of tags. Model accuracy was assessed by rank ordering these activations and choosing the model's highest N tags, where N is the number of tags used for that specific post. The model's chosen N tags were compared to the user's chosen N tags, and if a model's chosen tag was in the set of user's chosen tags, then the prediction was marked as correct. Otherwise, it was marked as incorrect.

This process was performed across all users in each of the 12 dataset slices for both the Twitter and Stack Overflow popular-users data sets. It was repeated for each of the decay rate values and the three different forms of B_i. For the hybrid form, three different values of k were explored (1, 5, 10).

Results

Model Performance Across Decay Rate Values

A plot of the results for one of the 12 slices, for both Twitter and Stack Overflow, and for all three forms of B_i are included in Figures 1 and 2, respectively.

Model accuracies at each decay rate value for each user in the dataset slice are included in the plots. Each user's mean accuracy was subtracted in order to compute a normalized mean for that user. This was done so that the relative accuracy change for different decay rate values could be more easily visualized. The plotting technique is analogous to what is done when testing the main effect in a repeated measures design (i.e., each subject's mean score is subtracted).

One large effect in all of the plots is that there is an optimal decay rate value between 0 and 20 that produces the highest model accuracy. For the standard B_i equation, model accuracy for a pure frequency model ($d = 0$) is also worse than using a pure recency model ($d = 20$), particularly for Twitter. It makes sense that a pure recency model for Twitter is relatively more accurate than for Stack Overflow, as the hashtag lifetime for Twitter is much less than a tag for a programming language on Stack Overflow.

When comparing the standard and optimized learning forms of the equation, it is apparent that a pure recency model does not work well for optimized learning. The best-fit decay rate value for the optimized learning form is also not as clear and pronounced as it is for the standard form (less of a peak). And the relative accuracy gained from using a pure frequency based model to a blend of frequency and recency is less for the optimized learning form than the standard form of B_i. Finally, for the range defined for the hybrid form ($d < 1$), the standard and hybrid forms are shaped similarly across the decay rate values.

Aggregate Model Performance

Aggregate best-fit decay rates were computed by taking the median of all user's best-fit decay rate values across all dataset slices. The median was used since there were users in the popular-users dataset that did not have enough observations to generate stable predictions, and best-fit decay rate values for these users could be as high as 20. Using the median effectively trims these unstable values from the sample, without having to define a cutpoint for an outlier removal process.

Aggregate best-fit decay rate values for each model are included in Figure 3. Decay rate values that produced the most accurate model performance are lower for the optimized learning form (0.43) of B_i compared to both the standard form (0.80) and hybrid form (0.80 for all k). The optimal values for optimized learning are near 0.5, which lines up with default value used in ACT-R.

Finally, model accuracy was analyzed across all dataset slices for Stack Overflow and Twitter. This was done by examining model accuracy at a specific decay rate value for each model. Accuracy was assessed at each model's previously determined best-fit values (0.43, 0.80, and 0.80 for the optimized learning, standard form, and hybrid form) and additionally the standard form's default value (0.5). Because decay rate values were explored at 0.1 increments between the 0 and 1 range, the decay rate values used to examine model accuracy were rounded to the nearest decay rate value that was included in the run. Consequently, a best-fit value of 0.4 was used for the optimized learning model instead of the 0.43 average found from the dataset slices. Aggregate accuracy was computed by taking the mean of every user's model accuracy across all users in all dataset slices.

Accuracy for both sites is included in Figure 4. Accuracy for the standard form is higher than the optimized learning form (.33 vs. 28, mean difference = 0.034, 95% CI [0.032, 0.036]). This makes sense given that the standard form does not assume equal presentation rate for each chunk, and hashtags on Twitter in particular show trends of high rate of use and then shift to low rate of use. The standard form of B_i is able to more accurately account for these dynamics.

Accuracy for the hybrid form for the largest k ($k = 10$) is slightly higher than the smallest k ($k = 1$) (.334 vs. 329, mean difference = 0.0026, 95% CI [0.0023, 0.0029]). However accuracy cannot be distinguished between the hybrid form with

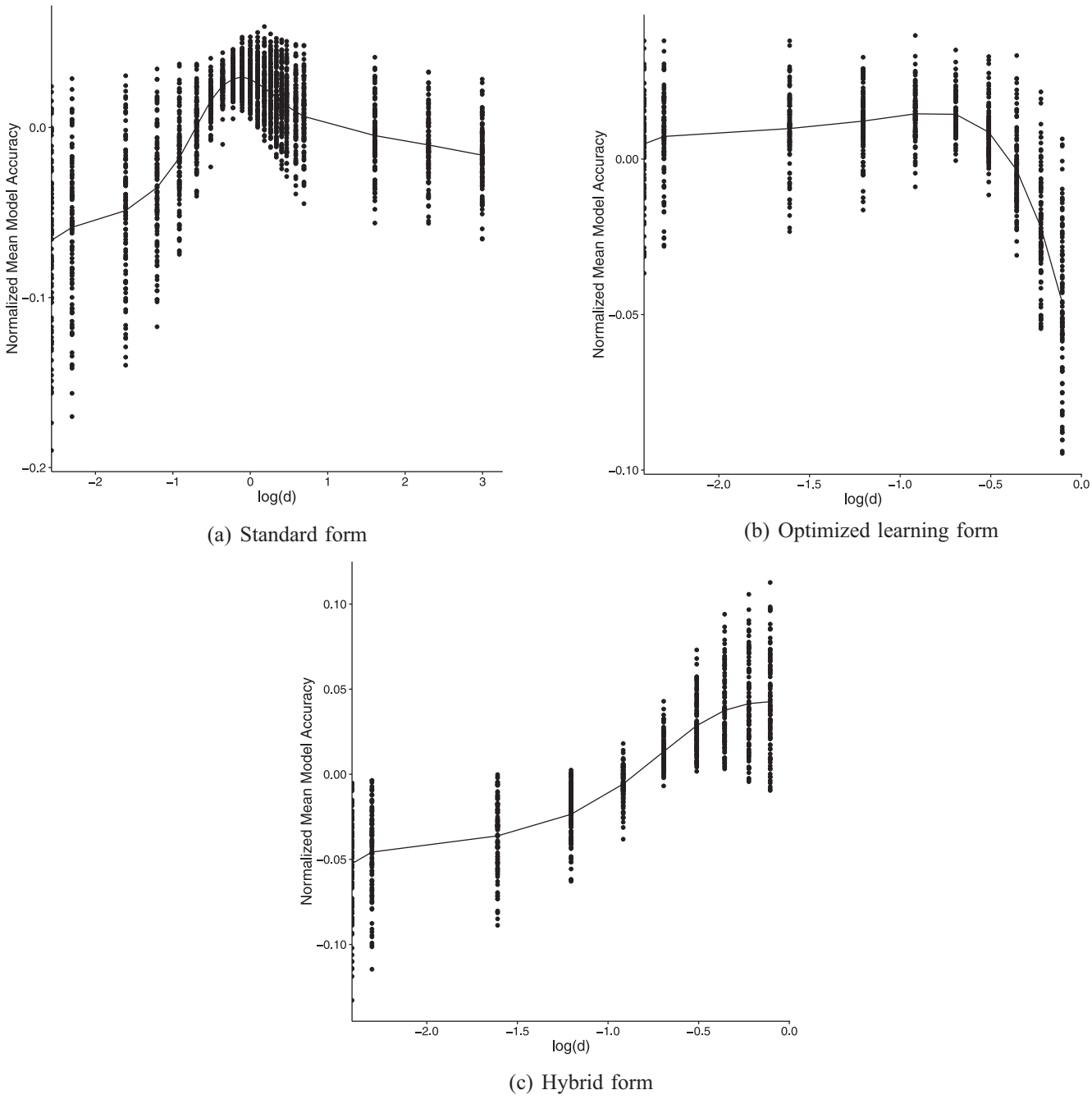
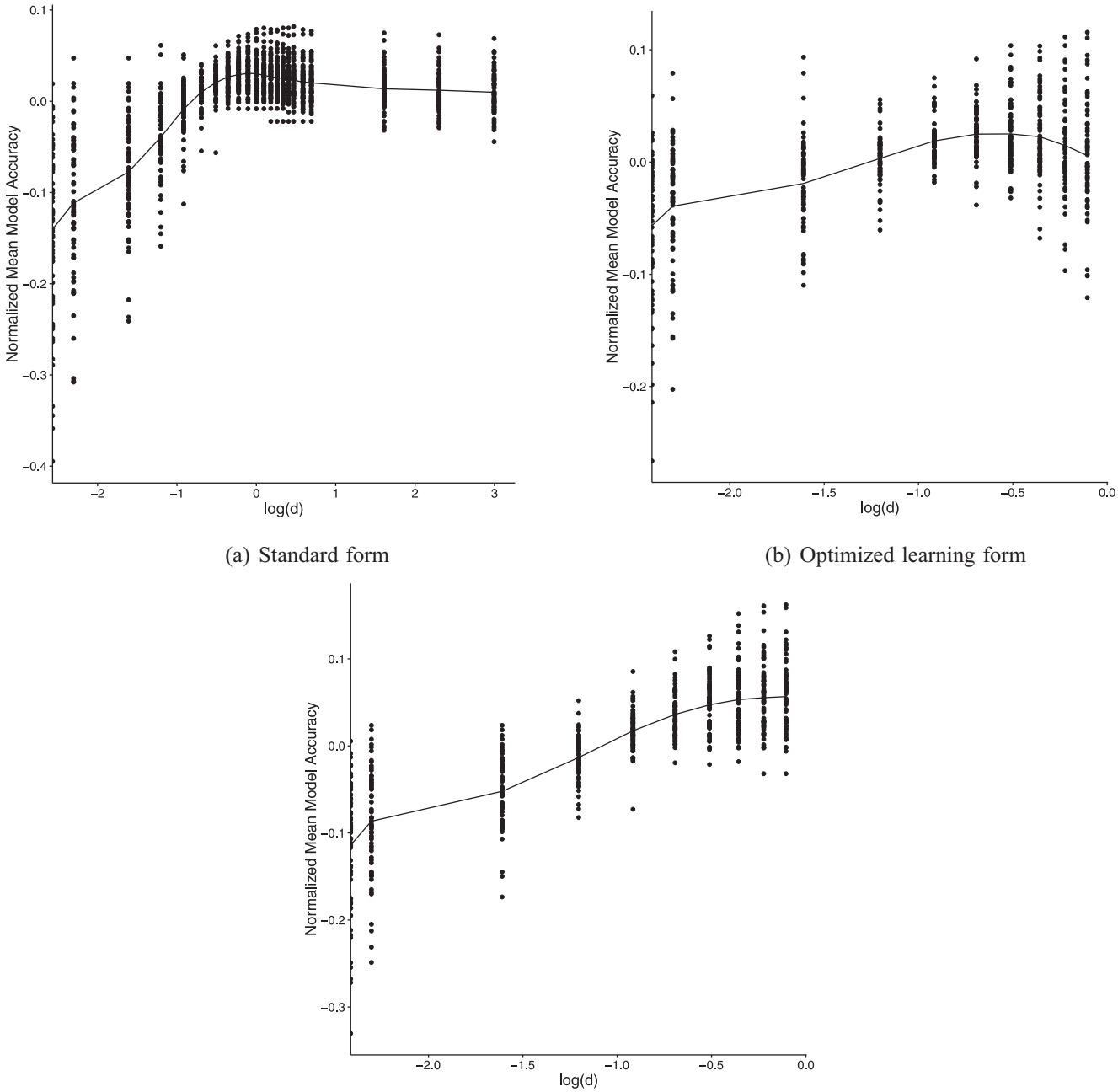


Figure 1. Model performance for a single dataset slice for Stack Overflow. This includes model accuracy as a function of decay rate value for each user in the dataset slice. Model accuracy is shown for the standard form of base-level activation (B_i), the optimized learning form, and the hybrid form when k = 10.

the largest k and the standard form (.334 vs. 334, mean difference = 0.0000, 95% CI [0.0000, 0.0001]). This indicates that for the data sets analyzed in this study the hybrid form with k = 10 is a suitable replacement to the standard form (produces the same results) and uses less computational resources than the standard form.

ACT-R sets the default decay rate value to 0.5, and the best-fit value for the standard form is higher (0.8) for these data sets. The figure shows that although model accuracy does slightly improve (0.325 to 0.335, mean difference = 0.0029, 95% CI [0.0022, 0.0036]) for the standard form when increasing the decay rate parameter from 0.5 to the optimal value, that percentage improvement is not as large (0.28 to 0.33) as the improvement gained from using the standard form compared to the optimized learning form. Nonetheless, model accuracy is highest when both the standard form is used and the decay rate value is increased.



(a) Standard form

(b) Optimized learning form

(c) Hybrid form

Figure 2. Model performance for a single dataset slice for Twitter. This includes model accuracy as a function of decay rate value for each user in the dataset slice.

Discussion

Given the large tag space used by post authors on Stack Overflow and Twitter, it was surprising that model performance is respectable even when only past tag history and no context is used. This speaks to the importance of taking into account overall past tagging history, and ensuring that the prior component of activation for each tag is customized to each user's specific past tagging history.

Also apparent for these data sets is that the optimized learning form of B_i is not as accurate as the standard form. Further, for the optimized learning form there is not much difference between using a pure frequency-based model and a model where frequency and recency are optimally blended. This makes sense given that the optimized learning equation in Table 4 collapses to a pure frequency model as the amount of time since the presentation of the first recorded chunk instance

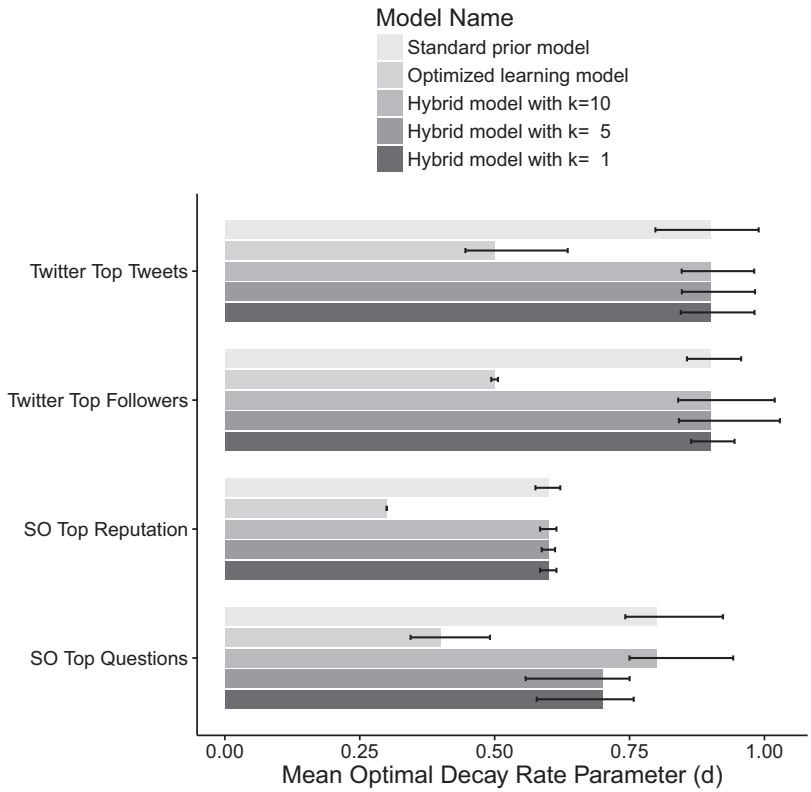


Figure 3. Overall best-fit decay rate for Stack Overflow and Twitter. Results are shown for the two Twitter and Stack Overflow (SO) popular-users data sets. Error bars represent the 95% bootstrapped confidence interval for the median.

increases, and also since the timespan of recorded tag use for Stack Overflow and Twitter is large (years).

However the hybrid form with a reasonable k is just as accurate as the standard form. This is due to the fact that the majority of change in activation for a chunk occurs in the short period of time after the presentation of the chunk (a property of a power law distribution). Once a chunk's presentations decay to the flat area of the power law distribution, not much change in activation for these presentations occurs, so they can be averaged and their presentation time can be discarded without much loss in fidelity. Therefore, if computational speed is a concern, one may wish to leverage the hybrid form instead of the standard form. Although qualitatively the standard form produces the cleanest and most natural effect of decay rate on accuracy, as shown in Figure 1. It also does not have a bifurcation at $k = 1$. Consequently, if one is a purist and computational speed is less of a concern, one may still wish to use the standard form over the hybrid form.

Combining Predictors

The declarative memory models analyzed for this research consist of two main components: past behavior and context. Past tag use was isolated in the first section to see how well ACT-R's prior component of declarative memory (i.e., base-level learning) fits each user's tag use over time. For the next section the past behavior and context declarative memory components were combined as well as looked at in isolation.

This section will examine how model accuracy is influenced by five architectural concerns: method of handling stop words, influence of dataset size, influence of compression for the random permutation model, method of combining prior and context, and word order. These particular architectural concerns were examined for the following reasons: During exploratory evaluation, model accuracy seemed to be highly sensitive to the method of dealing with the commonly used stop words, so several different methods were examined in more detail. Due to research from Budiu, Royer, and Pirolli (2007) and Recchia, Jones, Sahlgren, and Kanerva (2010), model accuracy for the Bayesian and vector-based models is influenced by the number of documents in the dataset, so the effect of dataset size on accuracy was examined in more detail. Sahlgren et al. (2008) showed how different levels of compression in the random permutation model influence accuracy, so this was examined for three reasons: to replicate the findings, as an additional check to ensure that the random permutation model was implemented correctly, and also to see if the same levels of compression are valid for the much larger data sets used in this research. One of the primary goals of this research is to identify the best method to combine the model term for past user behavior with the context term for the random permutation model, so two different ways to add the model terms were tested. Finally, previous research has shown that including word order in the vector-based models improves performance

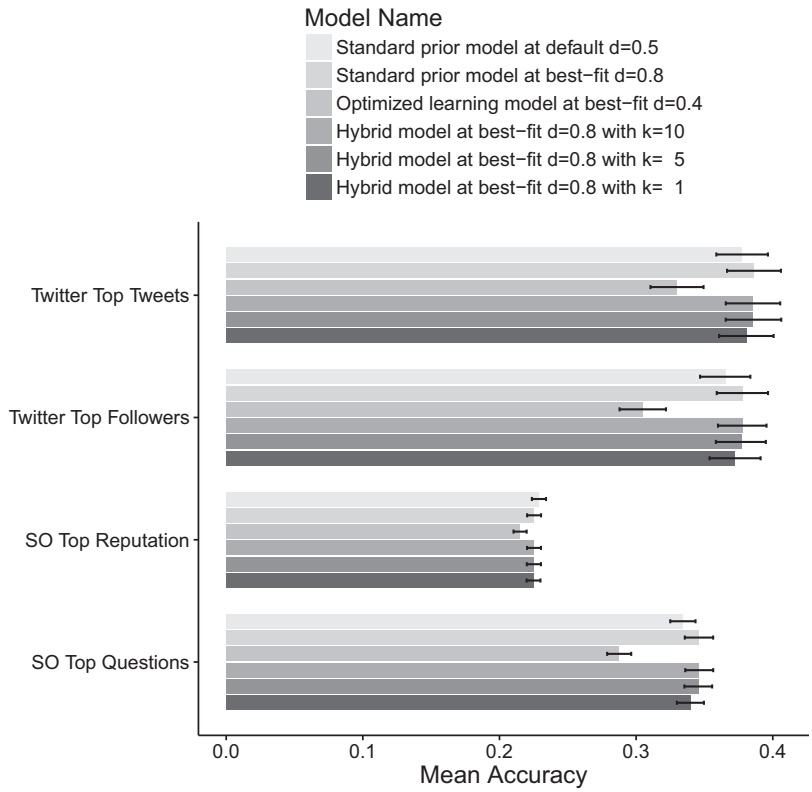


Figure 4. Overall model accuracy for Stack Overflow and Twitter at optimal decay rate values. All error bars are the 95% bootstrapped confidence interval for the mean.

(Jones & Mewhort, 2007; Sahlgren et al., 2008), so word order was also examined to see if the effect could be replicated and also to measure the overall importance of word order on model accuracy relative to other architectural manipulations.

Overall Method

Logistic regression. In order to best combine these predictors, a logistic regression statistical technique similar to Stanley and Byrne (2013) was used to determine the most optimal weights for each term. When a user creates a tag for a post, a retrieval request for the model is made. For each request, the model returns a set of tags (all tags that have been observed for that user in the past) and activations associated with each tag. An activation value is computed for each model component (e.g., prior tag use, context) for each tag. These values are recorded, and tag instances that match what the author actually tagged are marked as a 1, while all others are marked with a 0. When an author used multiple tags for a post, multiple 1s are marked for that post. This process is repeated across all posts where a user chooses a tag, and across all users in the dataset. Once all of the results are aggregated, a logistic regression is run where the optimal weights for each model component are found that maximize the model's ability to correctly label tags as 0 or 1 based on total activation.

Model accuracy was evaluated by rank ordering all recorded tags by model activation using the optimal weights, asking the model to tag the top *N* posts with a 1 (where *N* is the total number of recorded tagging instances in the dataset sample), and then comparing the model's chosen tags with the used tags for each post (i.e., looking at the "hits"). This is equivalent to throttling the threshold in the logistic regression such that the total number of observations labeled as a 1 match the total number of recorded instances of tagging in the dataset.

Adding offset term. This logistic regression method is a computationally efficient way to find the optimal weights for the predictors. However, when the regression cannot be constrained to label a specific number of 1s for each observation (as is the case here), to be used properly an offset term must be added as an additional model predictor. The offset term keeps the activation for the small number of top tags for each post equal across posts. This ensures that the logistic regression only labels a few 1s for each post.

A simplified version of Stanley and Byrne (2013)'s method was used for this research. Instead of subtracting the mean total activation for the top five tags for each post, the process is done in isolation for each model component. For the prior component for example, the activation value used in the logistic regression is the prior activation with the mean of the top five prior activations subtracted from this value. The same process is used for the context component. Decoupling the offset term and computing an offset for each model component in isolation means that an iterative approach was no longer necessary, since the optimal weights of the terms are no longer needed when computing the offset.

Data sets. Model accuracy was tested on the popular-hashtags dataset for Twitter and a random sample of posts across the entire Stack Overflow dataset. For each of the four Twitter popular-hashtags data sets, model retrievals were done across 15 runs and 500 posts within each run. For the Stack Overflow randomly sampled dataset, five runs of 500 posts were used, since the results for Stack Overflow were more stable and retrievals took more wall-clock time for this dataset due to the size of the co-occurrence matrix. All runs contained different posts, and all posts used for runs were not used to generate the co-occurrence matrix for the two retrieval models.

Computing priors. For each randomly sampled post for Stack Overflow, the prior activation is based on that user's prior tagging history, and not on any global prior of specific tag frequency across users. This was done so that the prior component used for Stack Overflow for both the popular-users and randomly sampled data sets was the same.

However, a custom user prior for the Twitter popular-hashtags dataset could not be used, because the entire Twitter dataset could not be downloaded, so the history of every user in the popular-hashtags dataset could not be collected. Instead, the global prior tag history for all users in the Twitter popular-hashtags dataset was used to compute prior activation values for each tag.

Co-occurrence matrices. The context component for both the random permutation and Bayesian model was based on the same co-occurrence matrix of words and observed tags. Different size matrices were analyzed, ranging from 1,000, 10,000, 100,000, 1,000,000, and 3,000,000 posts. The matrix for 3,000,000 posts was only used for Twitter, because the number of words in a Stack Overflow post is much higher than Twitter on average, and the space and computational time required when using a co-occurrence matrix of 3,000,000 posts for Stack Overflow was too large.

Sampling and overfitting. This research centers around evaluating the differences in model accuracy when various components are added, removed, or altered. When evaluating the models, overall model accuracy for a specific model and specific dataset is less important than how that accuracy deviates from a comparative model on the same dataset. Therefore, model overfitting is less of a concern in this case. Nonetheless, we did ensure that the models would not be substantially overfit in two ways: The posts used to build the co-occurrence matrices were never included as posts to calibrate weights of model terms. This is particularly important because the context term can easily latch on to a post with exactly the same co-occurrence structure as a post that was used to build the underlying matrix. Second, for each dataset sample, the logistic regression model that calibrated the model terms was fed with a large number of posts (500). Each model only had a few terms to calibrate (e.g., prior, context component), and there were on average 1,300 tags per sample that could be used by the logistic regression model for calibration, so the risk of overfitting was low. Further, Stanley and Byrne (2013) showed that the out-of-sample model and within-sample model accuracy difference for a similar model on a similarly sized dataset for Stack Overflow is small (2% accuracy difference). We were therefore more concerned with the risk of model overfitting by using fewer and more similar dataset samples than from optimizing the weights of a few model terms when those weights were calibrated from thousands of data points. This is why we focused on evaluating the models on multiple dataset samples (particularly the 60 different dataset samples used for Twitter) rather than to perform a k-fold cross-validation or out-of-sample validation technique for the weights from the logistic regression.

Stop Words

The method of handling stop words has been shown to influence model accuracy (Jones & Mewhort, 2007; Sahlgren et al., 2008; Stanley & Byrne, 2013), so this was the first architectural concern that was examined. "Stop words" (e.g., "the," "a," "or") are commonly used words that co-occur with all tags (i.e., have little to no discriminating power). Sahlgren et al. (2008) looked at several ways to handle the commonly occurring stop words for the random permutation model: A data-driven frequency-filtering approach was tried, along with using a previously generated stoplist, and no stop word removal at all. However, Sahlgren et al. (2008) did not try to weight the stop words and only tried methods to remove them.

Stanley and Byrne (2013) used the entropy weighting technique recommended in Dumais (1991) for the Stack Overflow dataset. The technique is formally described in Table 3 and uses information theory to determine the amount of predictive information contained in each word. Words that are often used with a small number of tags have high information content while words used often across all tags carry low information content. This method maps cleanly to the ACT-R DM theory by modifying each word's attentional weight (W_j) parameter.

Method. The runs from the Twitter popular-hashtags and Stack Overflow randomly sampled data sets were used for the analysis. Both the random permutation and Bayesian models were tested. Activations for context were computed by using five different methods to handle stop words: The two removal methods used in Sahlgren et al. (2008) (i.e., removal based on frequency, removal based on the predetermined 571-word Cornell SMART stoplist: Salton & Buckley, 1988), the entropy-weighting method used in Stanley and Byrne (2013), combining the entropy-weighting and frequency-filtering methods, and no removal or weighting of stop words.

Determining frequency cutoff. A cutoff point must be identified and used in order to remove the high-frequency words. Sahlgren et al. (2008) used a cutoff where words that occurred more than 15,000 times were removed. Because the Stack Overflow and Twitter data sets are much larger than the data sets used by Sahlgren et al. (2008), the same cutoff could not be used.

The ideal cutoff was determined by choosing the value where the standard deviation of the counts for each word for each of the tags were all relatively small. The plots for the standard deviations across tags that were used to identify the cutoff for Stack Overflow and Twitter are shown in Figures 5 and 6.

The "no processing" plot shows the size of the spikes when high-frequency words are not removed. The entropy plot shows how the size decreases by more than an order of magnitude after the entropy weighting measure is used. The cutoff for the frequency plot was chosen such that the plot looks similar to the entropy plot, with spikes around the same size. The cutoff that produced the frequency plots in Figures 5 and 6 is where a word represents more than 0.04% of total occurrences in the dataset.

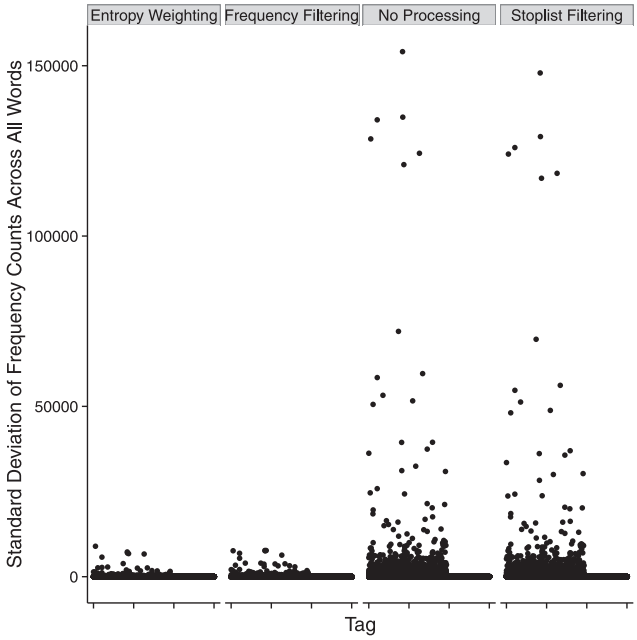


Figure 5. Variance in observation counts of words for each tag for Stack Overflow. Plots are shown for the four different methods for handling stop words. Each plot contains the standard deviation of the total counts for each word in context as a function of hashtag. High values mean that there are context words for a hashtag that have a high number of counts relative to the counts for all other context words for that hashtag.

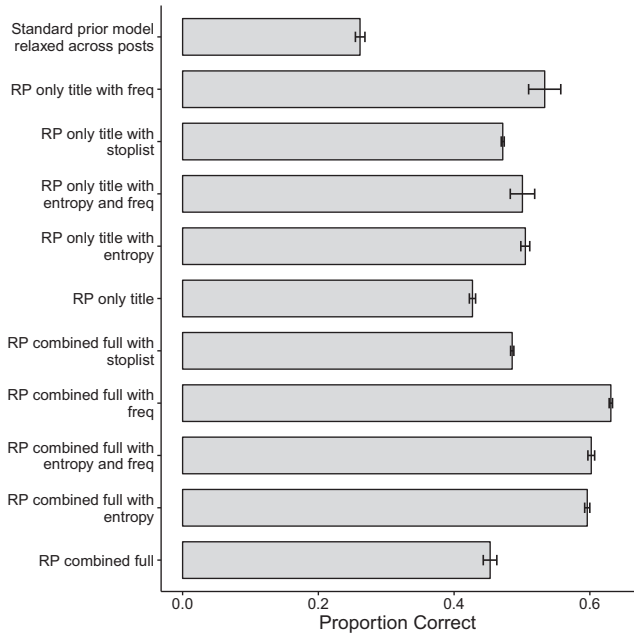


Figure 7. Stop-word techniques for the random permutation model for Stack Overflow. The five methods for handling stop words are shown both for the title model component in isolation and the full model. "Stoplist," "freq," and "entropy" are the stoplist removal method, removal based on frequency count, and weighting based entropy metric, respectively. All error bars are the 95% bootstrapped confidence interval for the mean.

Results for the random permutation model. Plots for the three methods of handling stop words for the random permutation model on the Stack Overflow dataset are shown in Figure 7. In the plot, if the model name includes "full" then all model components are included. When the model components are enumerated, then at least one of the components is left out. For example, the "standard prior model relaxed across posts" means that only the prior model component was used when determining activation. As another example, "RP only title with freq" means that only the title context component of the random permutation model was used, and that stop words were handled using the frequency filtering technique.

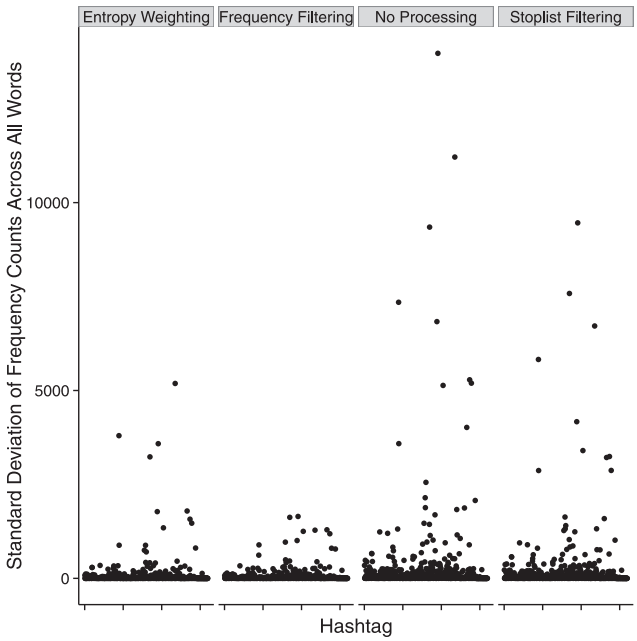


Figure 6. Variance in observation counts of words for each tag for Twitter.

The results show that model accuracy improves if stop words are handled in any of the three ways (predefined stoplist filtering, entropy weighting, and frequency filtering). Using a predefined stoplist makes the smallest improvement (.45 to .49, mean difference = 0.032, 95% CI [0.018, 0.047]) while the entropy (.45 to .60, mean difference = 0.143, 95% CI [0.123, 0.163]) and frequency (.45 to .63, mean difference = 0.178, 95% CI [0.164, 0.192]) techniques improve performance the most, and frequency shows a slight edge over entropy. However, using the frequency-filtering technique produces more noise in accuracy measurements compared with the other techniques, at least when tested on words in the title of Stack Overflow posts.

The three stop word techniques were then tested for the random permutation model on the Twitter dataset, and the results are depicted in Figure 8. For the random permutation model for Twitter, the entropy weighting technique is the best method for handling stop words (.25 to .26, mean difference = 0.013, 95% CI [0.011, 0.015]). Removing stop-words based on a predetermined list actually decreases accuracy (.25 to .23, mean difference = −0.017, 95% CI [−0.015, −0.019]), and

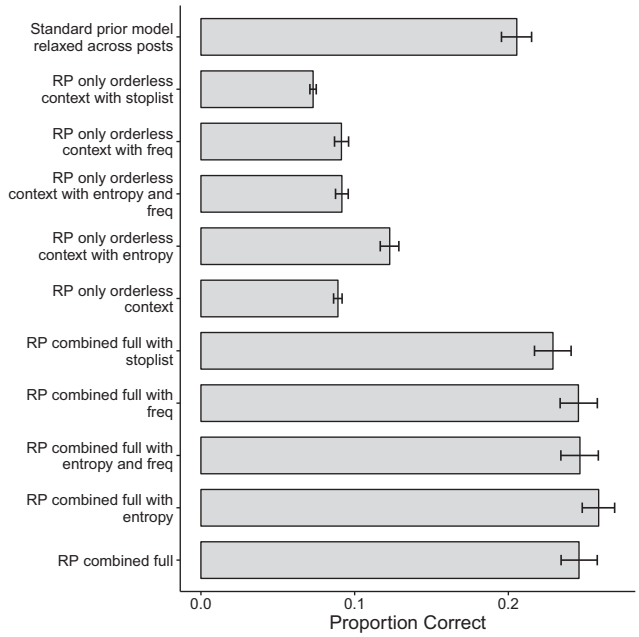


Figure 8. Stop-word techniques for the random permutation model for Twitter. The five methods for handling stop words are shown both for the context model component (i.e., no prior term) and the full model. All error bars are the 95% bootstrapped confidence interval for the mean.

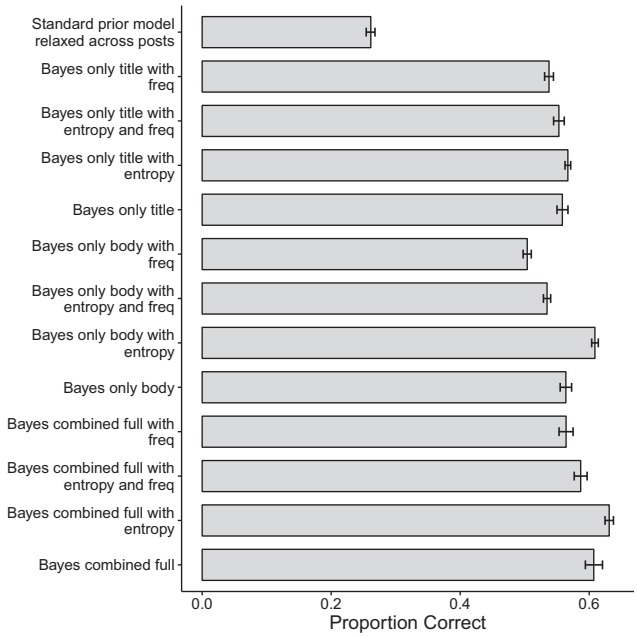


Figure 9. Stop-word techniques for the Bayesian model for Stack Overflow. All error bars are the 95% bootstrapped confidence interval for the mean.

using data-driven frequency filtering has only a small to no effect (.25 to .25, mean difference = 0.00, 95% CI [−0.003, 0.002]).

Results for the Bayesian Model. Because accuracy measurements when using a predetermined list to remove stop words for the random permutation model were much worse than the frequency and entropy methods, this method was not examined for the Bayesian model. The results for the frequency and entropy techniques for the Bayesian model on the Stack Overflow dataset are depicted in Figure 9.

The results show that entropy weighting produces the most accurate results (.61 to .63, mean difference = 0.024, 95% CI [0.011, 0.037]). Also, accuracy actually decreases when using frequency filtering compared with no filtering (.61 to .56, mean difference = −0.043, 95% CI [−.030, −0.057]) for the Bayesian model. Therefore, entropy weighting is the clear winner in this case.

Similar results are found when the Bayesian model is tested on the Twitter dataset. Figure 10 shows that entropy weighting is again the best performer (.29 to .32, mean difference = 0.028, 95% CI [0.024, 0.031]), and frequency filtering decreases performance relative to no filtering (.29 to .28, mean difference = −0.017, 95% CI [−0.015, –0.019]).

Discussion. One result that should not be overlooked is how poorly using a predetermined stoplist performed when compared to the two data-driven frequency and entropy approaches. This is most likely because the Stack Overflow and Twitter data sets are large and contain high-frequency words that are domain specific (e.g., emoticons) and are not included in the predetermined list. This shows that data-driven techniques to identify stop words can be much more effective than using a predetermined list.

Another interesting result is how the Bayesian model actually performed worse when stop words were removed based on frequency compared with no stop word removal. This was not the case for the random permutation model in this study, was not the case when Sahlgren et al. (2008) used frequency filtering on the random permutation model, and is most likely never the case

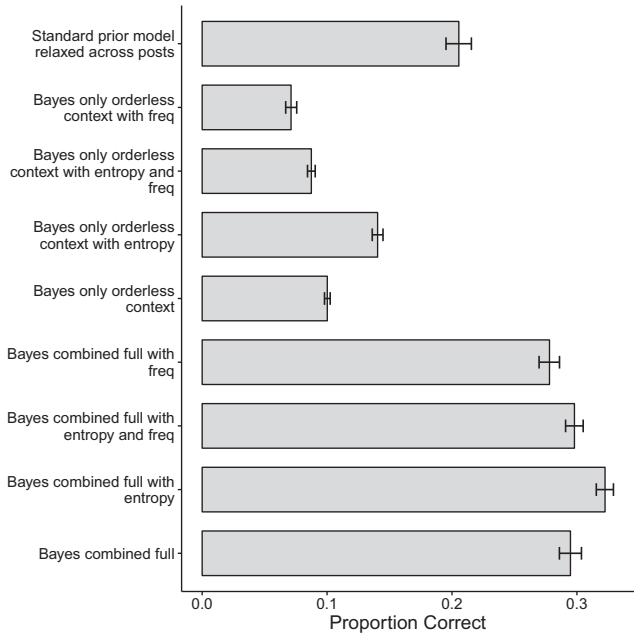


Figure 10. Stop-word techniques for the Bayesian model for Twitter. All error bars are the 95% bootstrapped confidence interval for the mean.

for the random permutation model. Model accuracy most likely decreases for the Bayesian model because this model already has a normalization component built in that the random permutation model does not have. When computing activation for the random permutation model, the correlation is calculated directly on a matrix of counts of co-occurrences for words and tags. For the Bayesian model this count matrix is converted to a log-odds (S_{ji}) matrix, where both the number of observations for the word and the tag are normalized when computing the value for each cell (see the S_{ji} computation in Table 3). Therefore, it is likely the case that a form of frequency attenuation is already built into the Bayesian model, and adding a frequency filter on top of this does nothing to improve accuracy and only removes predictive information.

Stanley and Byrne (2013) found that the entropy-weighting technique to attenuate stop words for the Bayesian model significantly improved model accuracy. The same effect is found for the Stack Overflow and Twitter data sets in this study. Further, when this entropy-weighting technique is compared directly with the more common frequency-filtering technique, weighting by entropy produces more accurate results. This was clearly the case for the Bayesian model. Both frequency and entropy techniques performed equally for the random permutation model. However, since the entropy-weighting technique is parameter-free and does not require a cutoff to tune, it is also a better choice for the random permutation model for these data sets.

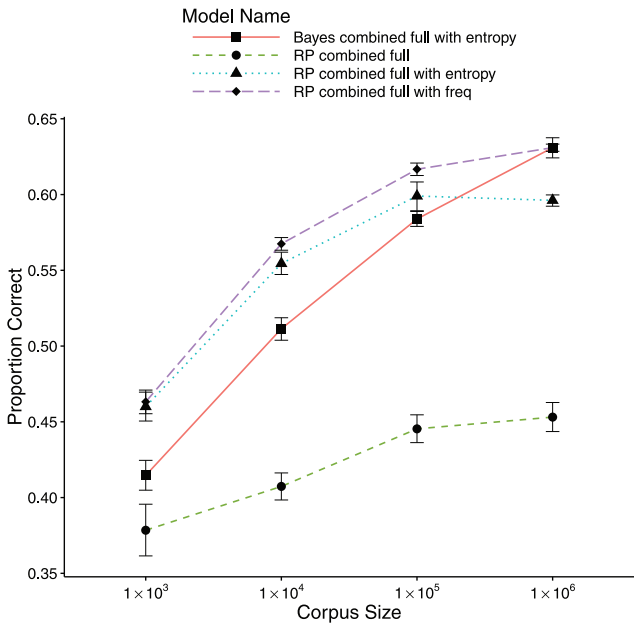


Figure 11. Effect of size of co-occurrence matrix for Stack Overflow. Error bars represent the 95% bootstrapped confidence interval for the mean model accuracy across all runs in dataset. See the online article for the color version of this figure.

Corpus Size

The number of documents used to build the co-occurrence matrix has been shown to influence model accuracy (Sahlgren et al., 2008). The default number of documents used to build the co-occurrence matrix was 1,000,000 for Stack Overflow and 3,000,000 for Twitter. In order to see how accuracy changed as function of corpus size, co-occurrence matrices for a range of smaller corpus sizes were built.

Method. For Stack Overflow, the following number of posts were used to build separate co-occurrence matrices: 1,000, 10,000, 100,000, and the default 1,000,000. For Twitter, these matrix sizes were tested alongside the default 3,000,000. All runs from the four Twitter popular-hashtags data sets and the randomly sampled Stack Overflow dataset were used to test model accuracy.

Results for Stack Overflow. Model accuracy as a function of co-occurrence matrix size for Stack Overflow is depicted in Figure 11. Accuracy improves for both the Bayesian and random permutation models as the size of the documents used to generate the co-occurrence matrix increases. However for the random permutation model, accuracy begins to plateau for the largest corpus size for all three tested versions. The Bayesian model does not plateau nearly as much and accuracy continues to improve even at the largest measured corpus size.

Results for Twitter. The overall trend of increasing accuracy with corpus size is present for Twitter as well. Both entropy weighting and frequency filtering in Figure 12 plateau for the random permutation model with increasing corpus size. Once again, the Bayesian model continues to improve with corpus size.

Discussion. It is interesting to compare model accuracy between the Bayesian and random permutation model at the smallest and largest corpus sizes for both Stack Overflow and Twitter. At smaller corpus sizes, the random permutation model outperforms the Bayesian model (slightly for Twitter, dramatically for Stack Overflow), and this performance difference diminishes as corpus size increases. In other words, the Bayesian model needs larger co-occurrence matrices than the random permutation model to work properly. As the size of the dataset increases, the compression may start to lose crucial information, and the uncompressed Bayesian model should be used. As the size of the dataset decreases, the noise and crosstalk introduced by the compression for the random permutation model is actually beneficial, and the compressed random permutation model should be used. This noise is beneficial because it helps soften spurious S_{ji} values when the counts for each individual cell in the co-occurrence matrix are still volatile, which happens at smaller corpus sizes.

This result has important implications. The corpus size is rarely chosen, and more likely constrained by the number of documents in the dataset being studied. If that dataset is relatively small, it may be better to use the compressed random permutation model than the Bayesian model, and vice versa if the dataset is large.

However, corpus size is not the only factor that determines whether the Bayesian model or random permutation model will perform better. This is because the number of observations used to generate the co-occurrence matrix is a function of four components: corpus size, number of words in each document, total number of unique tags, and number of tags used per document.

The compressed random permutation model should perform better with smaller corpus sizes, less words per document, more unique tags, and fewer number of tags per post. However, because there are so many factors that determine whether the Bayesian or random permutation model will perform better, and only two different data sets were examined in this research, it is difficult to make any specific recommendations as far as exact transition cut

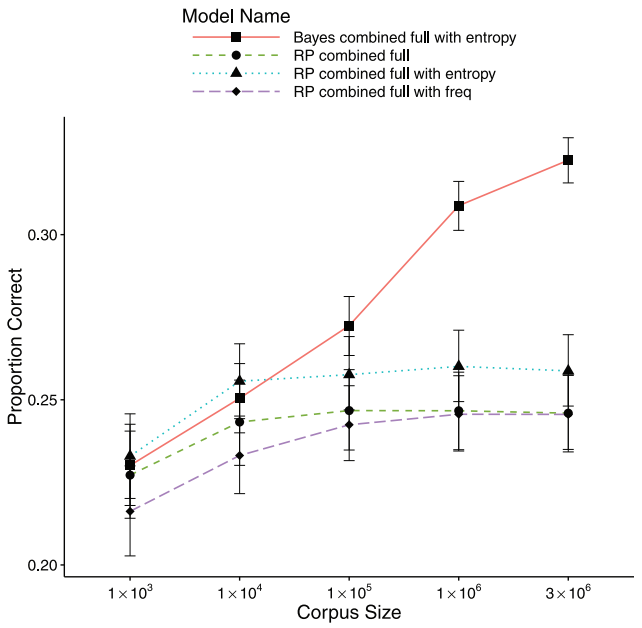


Figure 12. Effect of size of co-occurrence matrix for Twitter. Error bars represent the 95% bootstrapped confidence interval for the mean. See the online article for the color version of this figure.

points (e.g., corpus size, words per document) where either the Bayesian or random permutation model should be used. Also, it is likely that the specific domain being studied interacts with the factors already mentioned to determine which model is better for that domain. Nevertheless, if a researcher is working in a domain with limited data (e.g., small corpus size, large tag space) that produces a sparse co-occurrence matrix, then it may be worthwhile to experiment with using a compressed random permutation model instead of the full Bayesian model. Also, newer models of S_{ji} have been introduced that have a more local impact when a new chunk is observed (Thomson, & Lebiere, 2013). It would also be interesting to see how these local models of S_{ji} perform while varying the number of observed associations.

Compression Level for Random Permutation

Another parameter in the random permutation model is the number of rows used to represent all of the different words. Each word's environment vector is represented by two randomly placed ones and two negative ones across the rows. All that is required is that the four positions and signs of each of these environment vectors is unique. But this means multiple environment vectors can overlap when looking at a single row. This is a lossy form of compression, and allows the number of rows to represent the words to be much less than the total number of words. As the number of rows decreases, the amount of crosstalk between the environment vectors increases, compression increases, and the amount of information stored decreases. As a third architectural concern, the effect of compression for the random permutation model was examined for Twitter and Stack Overflow.

Method. Three different row-size values were tested for the random permutation model to see how accuracy changed as a function of space required to represent the co-occurrence matrix. A low value (100), a more standard value (2,048), and a high value (10,000) of rows were tested. The 2,048-row matrix was the default size used for the random permutation model for all other results. A separate co-occurrence representation was built for each of these row values. Each representation was tested on the four Twitter popular-hashtags data sets and the Stack Overflow randomly sampled dataset. Accuracy measurements were the same as were used for previous runs of these data sets: mean accuracy for each of the five Stack Overflow and 15 Twitter runs.

Results. Results for using different levels of compression for the random permutation model for the Stack Overflow dataset are depicted in Figure 13. The plot is also broken out for different size data sets used to build the co-occurrence matrix.

Reducing compression to only 100 rows in the representation has a strong negative effect on accuracy, which shows that compression can influence model accuracy. However, 2,048 rows is sufficient for these data sets, as increasing the number of rows to almost an order of magnitude higher (10,000) has little to no positive impact on accuracy, even for the highest corpus sizes.

Results are similar for Twitter. Figure 14 shows the same negative impact when reducing from 2,048 to 100, and little to no impact when increasing from 2,048 to 10,000. These results are consistent across all corpus sizes used to build the co-occurrence matrix.

Discussion. The fact that the random permutation model only needs around 2,048 rows to represent all of the words in the Twitter and Stack Overflow data sets is impressive. Twitter has around 2,383,000 unique words for the 3,000,000 tweets in each of the popular-users data sets. Stack Overflow has 5,848,000 unique words for the 1,000,000 posts used to generate the co-occurrence matrix. This means that only 2,048 rows are needed to represent

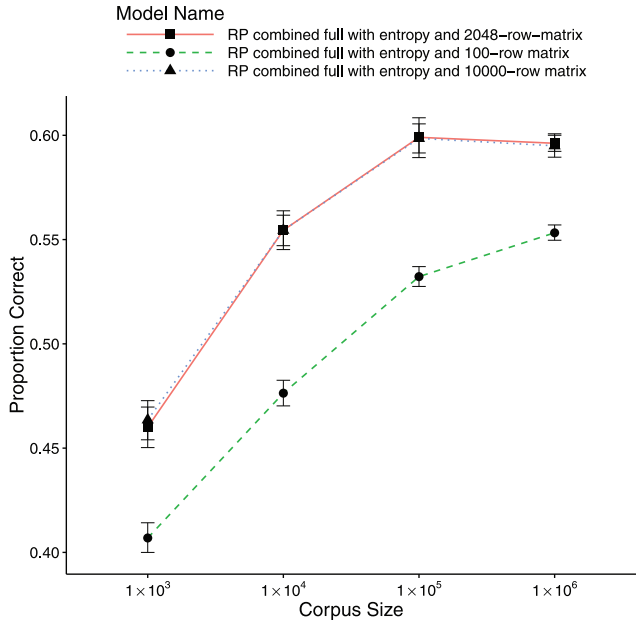


Figure 13. Effect of compression on random permutation model for Stack Overflow. Error bars represent the 95% bootstrapped confidence interval for the mean. See the online article for the color version of this figure.

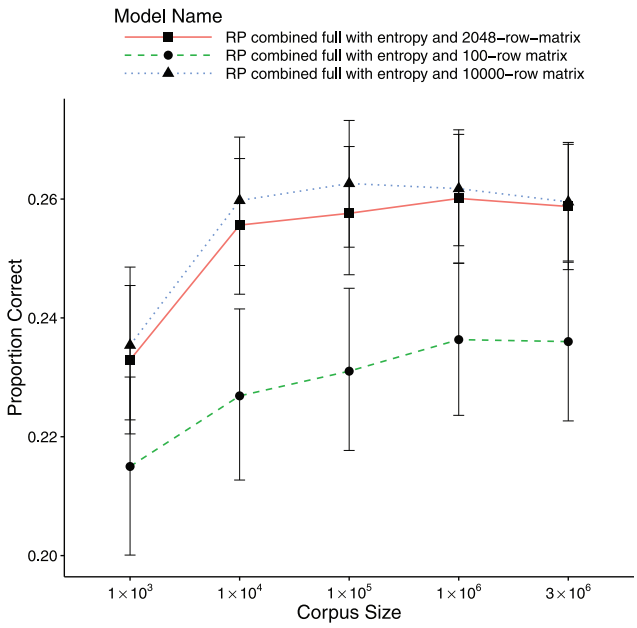


Figure 14. Effect of compression on random permutation model for Twitter. Error bars represent the 95% bootstrapped confidence interval for the mean. See the online article for the color version of this figure.

most of the variance from all of the words in these co-occurrence matrices. This is likely because word use frequency for Stack Overflow and Twitter approximately follows Zipf's law. Stanley and Byrne (2013) showed that frequency of tag use for Stack Overflow approximately follows a Zipf's law distribution. This means that if the words are rank sorted by number of times they are used, the frequency of word use drops off sharply. Because these low-use words have little to no ability to influence predictions, it makes sense that accuracy does not substantially improve by providing additional dimensions for them.

Combining Prior and Context

The activation value for the context component of the vector-based model is a correlation, ranging from zero to one. This is not a log-odds value. For the previous results, prior and context for the random permutation model were combined by simply adding the terms. However, this means that ACT-R's prior base-level learning component (which is a log-odds value) is added to a correlation value, which may not be mathematically appropriate. As a fourth architectural concern, another method to combine the prior and context components for the random permutation model was explored.

Method. For a given retrieval, the random permutation model returns a set of activations that are correlation values for each of the tags. Instead of simply adding this correlation value to the prior model term, it may be more appropriate to convert the correlation value to a log-odds value first, and then add this context term to the log-odds prior term. So a method was created to convert a distribution of correlations to a distribution of log-odds values. To do this, the one-tailed cumulative distribution function value for each correlation was computed. That probability was then converted to a log-odds value in the usual way.

This technique has a few advantages: It is simple and computationally cheap. Also, it behaves properly at specific points in the distribution. For example, if the conversion is performed on the mean correlation value in the set, the result is a log-odds value of zero. This makes sense given that log odds can be interpreted as the amount of odds information (positive or negative) that should be adjusted to the prior log odds, given that new piece of information. Because the mean correlation value contains no information regarding if it is better or worse than the other correlation values, it is appropriate to assign this correlation value a zero log-odds adjustment value.

The log-odds transformation technique was tested for the random permutation model on the Twitter popular-hashtags and Stack Overflow randomly sampled data sets.

Results. Model accuracy results after using the log-odds transformation are depicted in Figures 15 and 16. Using the log-odds transformation technique has a small effect on model accuracy for Stack Overflow (.45 to .47, mean difference = 0.015, 95% CI [0.009, 0.021]) and minimal to no effect on Twitter (.25 to .25, mean difference = −0.002, 95% CI [−0.005, 0.001]).

Discussion. Vector-based models such as the random permutation model (Sahlgren et al., 2008) have included only a context component in the past (i.e., no prior component). When the prior component is included for the random permutation model, accuracy improves significantly (compare accuracy for full models to context models in Figures 15 and 16). However, it is somewhat surprising that it does not matter much how the context and prior terms are added for the random permutation model. Model performance is similar when the context component used for the model is based on a correlation compared with a log-odds value.

There is a slight edge in increased accuracy when a log-odds transformation is used for the Stack Overflow dataset, although the

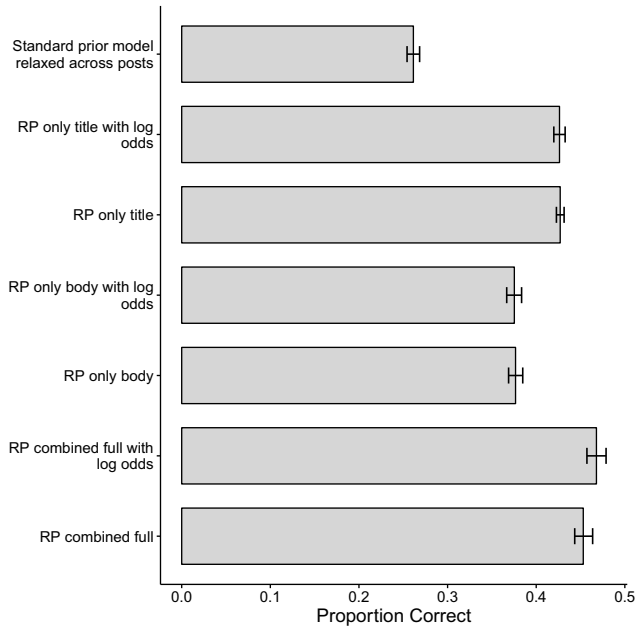


Figure 15. Random permutation model accuracy when using log-odds transformation on context for Stack Overflow All error bars are the 95% bootstrapped confidence interval for the mean.

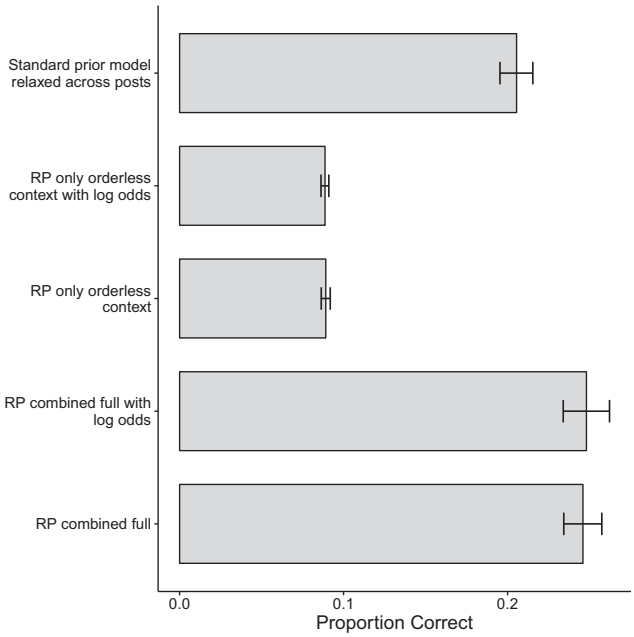


Figure 16. Random permutation model accuracy when using log-odds transformation on context for Twitter. All error bars are the 95% bootstrapped confidence interval for the mean.

magnitude of the improvement in accuracy is small (.45 to .47). Nonetheless, it does seem appropriate to convert the context to a log-odds adjustment value when context is added as an additional term to the random permutation model. Because performance does not decrease when the transformation is made, may slightly increase, and leads to a more natural interpretation of each model term as a log-odds value, it is reasonable to include this step for the random permutation model. Future research could explore additional methods to properly combine the terms.

Word Order

As a final architectural concern, vector-based models with and without word order were tested to examine the effect of word order on model accuracy. Vector-based models like the random permutation model can easily incorporate word order when making predictions. This has been touted as one of their primary strengths (Jones & Mewhort, 2007; Sahlgren et al., 2008). In order to thoroughly test the impact of adding word order into the model, various methods for incorporating word order were tested on the Stack Overflow and Twitter data sets.

Method. Three different methods for incorporating word order in the random permutation model were tested: taking just the sign of the word's position relative to the tag (direction method), using the position (standard method), and using the position but only including words that were used near the tag (window method). For the standard method, all words and their respective position are used to build the representation. For the direction method, essentially two representations are created: all co-occurrences with words to the left of the tag, and all co-occurrences with words to the right of the tag. The window method is similar to the standard method, where each word's relative position is maintained, but it only includes words near the tag to build the representation. Sahlgren et al. (2008) tested similar methods, and found that accuracy was highest when a narrow window was used for the window method (only including words at most two positions to the left and right of the tag). The same $+2$ -2 range was used to test the window method on the Twitter dataset.

Only the Twitter dataset was tested, because the Stack Overflow dataset contains no word order information between words in the title and body of the post and the tags used for the post. The author chooses tags after filling out the title and body forms in the post, so the tags are never intermixed with words in the post like they are for Twitter.

The same four Twitter popular-hashtags data sets were used for testing. Both an ordered and orderless context component were created for the random permutation models. The orderless context component uses the same co-occurrence matrix used by the Bayesian model to create the representation, because the Bayesian model does not contain any order information. Models were tested that used only one of the three word order methods for the ordered context component, used only the orderless context component, used a combination of the two components, and combined context with prior.

Results. Model accuracy results after incorporating word order into the random permutation model are depicted in Figure 17. All bar plots that include an order component where "window" or "direction" is not included in the title use the standard ordering method (no direction or windowing, all position information is included). This is the default ordering method used, so the names for these bar plots were kept consistent to the names used in other figures so that comparisons could be made across figures. Base

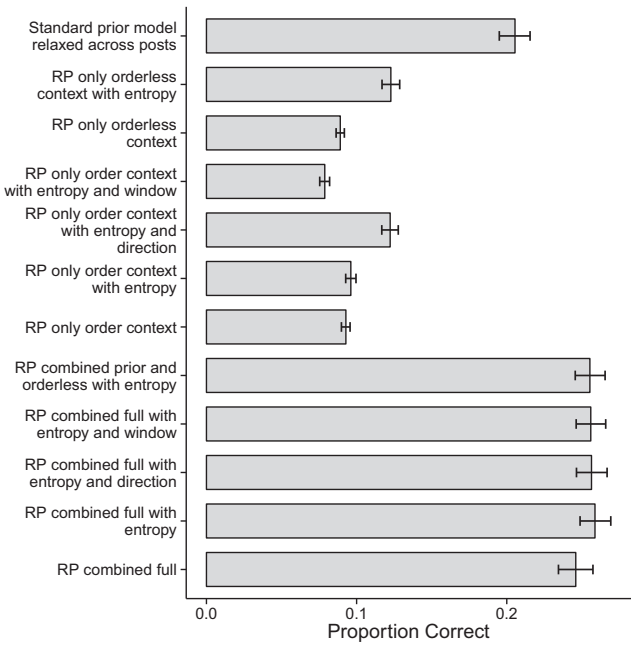


Figure 17. Model accuracy for various word order methods for the random permutation model for Twitter. All error bars are the 95% bootstrapped confidence interval for the mean.

models that do not use entropy weighting were also included so that effect sizes could be compared between entropy weighting and adding word order information.

Looking first at models where no prior is added, the random permutation model using only orderless context performs similar to the model using only order context with direction. All other order-only models perform worse than the order model using direction and worse than the orderless model.

Full models were also plotted against the model where only the prior and orderless context was included (i.e., no order). This was to test how much of the variance in model accuracy is shared between the orderless and order models. The results show that there is little unique variance in the order-only models, so order information contributes little to overall model performance above and beyond what an orderless model contributes. This can be seen by comparing the two-component model with only prior and orderless context to the three-component model with prior, orderless, and any of the three methods for the order context term.

Discussion. This was a surprising result. One of the main benefits touted for the random permutation model is that it can easily incorporate word order information into the representation. But including word order information does not dramatically improve performance. For the random permutation model, it is much more important to properly handle low-predictor stop words and include a prior component than it is to include word order information, at least for the Twitter dataset tested in this study.

The original BEAGLE model (Jones & Mewhort, 2007) actually showed only minimal improvement when incorporating word order as well. In absolute terms, this model only improved by two percentage points in accuracy when trained on the TASA and tested on the TOEFL. Sahlgren et al.'s (2008) results for word order for the random permutation model are similar. At a row size of around 2,000, model accuracy improves by less than a percentage point when order information is incorporated.

Although incorporating word order into the random permutation model does seem to slightly improve performance, the strength of the model is more likely to be in how efficiently it can represent the co-occurrence matrix rather than its ability to include word order in the representation. That is, the strength of the model is in its compressed unordered context component term and not the ordered component, at least for the Twitter dataset studied.

Also, the reason why the directional version of ordering is the best predictor of the three ordered components is most likely because order information does not have a large effect, so the two representations for words to the left and right of a tag are highly correlated. This means that this directionally ordered component is essentially composed of all the co-occurrence observations in the orderless component after being randomly sampled and partitioned into two separate representations. The co-occurrences in these representations are highly correlated with each other, and highly correlated with the orderless component, which is why model accuracy is similar for the orderless component and direction ordered component, and why accuracy does not improve when adding the direction ordered component to the orderless component.

General Discussion

The general discussion is separated into two parts. The first section will describe the overall results, and the final section will go through each key finding and discuss theoretical implications and future directions for research.

Overall Results

Table 5 summarizes model accuracies for all models and data sets tested in the study. This includes the prior-only model tested on the popular-users dataset and the full models (context and prior) tested on the popular-hashtags and randomly sampled data sets. After modifying the models by adding an entropy weighting mechanism to both models and a prior term to the random permutation model, accuracy for the compressed random permutation model was nearly equal to the Bayesian model. The largest difference between the two models was on the Twitter popular-hashtags dataset (7%). Both models performed nearly equal for the Stack Overflow data sets tested, and both models were much more accurate when tested on Stack Overflow compared with Twitter.

Implications and Future Directions

Tagging as a declarative memory retrieval process. Overall, both the random permutation and Bayesian declarative memory models fit the observed tagging behavior in Stack Overflow and Twitter reasonably well. The fits for Stack Overflow in particular are quite high, especially given the large tag space on the site.

These results provide support for the idea that choosing a hashtag when composing a tweet and tagging a Stack Overflow post is akin to a declarative memory retrieval process. However, both models are not 100% accurate, and the amount of variance

Table 5
Summary of Accuracy for Each Model and Dataset

Site	Dataset	Model	Accuracy
Stack Overflow	popular-users	Standard prior model	.28
	randomly-sampled	Bayesian with entropy	.63
		RP with entropy and log odds	.63
Twitter	popular-users	Standard prior model	.37
	popular-hashtags	Bayesian with entropy	.32
		RP with entropy and log odds	.25

Note. The Bayesian model includes entropy weighting for the context words, the RP model includes entropy weighting and log odds transformation when combining the context and prior components.

left to predict is larger than what would be observed if retrieval noise is included in the model. One possibility that seems likely is that the tag selection process includes other processes besides just declarative memory retrieval. For example, it may be the case that the declarative system is used to provide a set of likely tag candidates to higher-level processes to consider for selection.

Another possibility is that this is not a declarative memory process at all. In particular, it is important to note that tagging on these sites is widely used as a method to communicate specific messages to specific groups of people in the community. It is unclear how much this environment overlaps with verbal communication between two people, and the declarative memory requests required to successfully communicate in this setting. Nonetheless, these models are reasonably accurate in this setting, particularly for Stack Overflow. Given the results in this research, it at least seems worthwhile to use these publicly available tagging environments as a reasonable proxy to explore and validate modifications to the current theory.

Representation. Given that the storage space required to represent the co-occurrences for the random permutation model is dramatically smaller than the uncompressed Bayesian model, and that the compression is lossy and essentially random (i.e., no SVD), it is quite interesting that the random permutation model performs nearly as well as the full Bayesian model for this task.

These results pose an interesting and important fundamental question about declarative memory: How, precisely, is the neural system configured to represent co-occurrences for declarative memory? Because at the behavioral level both models perform similarly, it is possible that the neural system is using something similar to either representation to store co-occurrences. Vector-based models are simple enough that they can easily be implemented in a neural network. Perhaps a form of vector-based compressed storage is being used at the neural level, because it can be implemented in neurons and is an efficient way for the machinery to generate predictions that closely match the statistical structure of the environment (i.e., that closely match the uncompressed Bayesian representation).

Compression. As previously noted, compression for the random permutation model is essentially random. That is, no SVD is done to identify the most predictive components, so the compressed representation used by the random permutation model does not contain nearly as much of the variance from the original co-occurrence matrix than it could have. However, performing a full SVD on a co-occurrence matrix as large as the ones generated for Stack Overflow and Twitter is computationally infeasible. Also, when deriving the LSA approach, Landauer and Dumais (1997) explicitly state that they make no claims that the brain is computing an SVD at the neural level; only that the neural machinery is set up such that the representation makes predictions that are similar to what would have been generated if the matrix decomposition for an SVD would have been computed. Perhaps the randomly compressed representation used for the random permutation model is the way in which the brain tries to efficiently achieve the full LSA- and SVD-like approach that Landauer and Dumais propose.

Word order. The relatively strong performance of the random permutation model is not due to the model's ability to represent word order. Performance is similar for the random permutation and Bayesian models when the co-occurrence matrix for both models is the same, and not based on word order. Performance also does not improve by much when word order is added as the last predictor in the random permutation model. The random permutation model seems to work so well mainly due to the way that the full co-occurrence representation is compressed.

Corpus size. The random permutation model was more accurate than the Bayesian model at smaller corpus sizes, while the results were flipped for larger document sizes. This is interesting from a practical standpoint because the corpus size is rarely chosen in declarative memory research, and is constrained mainly by the amount of data available in the domain being studied. If a researcher is working within a domain that has a relatively small corpus size (generally near or less than 1,000 documents), it may be worthwhile to test the random permutation model in this domain, as this compressed model may be more accurate than the uncompressed Bayesian model.

Further, it is interesting to study the effect of model accuracy on corpus size because people require exposure to only a small number of documents before tagging documents appropriately. On Stack Overflow, each user asked an average of five questions. Model accuracy was high for Stack Overflow, but the model required training on one million documents before achieving that accuracy. The user only needed to "train" on five questions and the model needed to train on one million questions.

This is a large discrepancy, but this does not necessarily invalidate the declarative memory models used for this research. First, the random permutation model was still accurate at smaller document sizes, so perhaps this result provides evidence to favor the random permutation model over the Bayesian model. However, the smallest corpus size tested for these models was 1,000 documents, which is much greater than the average number of questions or tweets that a user contributes to the site.

Second, the models may require large amounts of training because the co-occurrence matrices are being built from scratch. That is, the models are being trained on the Stack Overflow and Twitter domains without any prior general co-occurrence information to start with. Presumably, when a person starts using a site, they can leverage all of their past exposure to different word co-occurrences and then simply add on top of that base the domain-specific co-occurrences. In that sense, one could argue that the models are undertrained on generic co-occurrences (people are exposed to many more than one million documents in their lifetime) and overtrained on specific co-occurrences. It may be interesting to test these memory models on a hybrid co-occurrence matrix, where the majority of the observations are taken from a generic domain (e.g., American literature) and a minority are domain-specific (e.g., Stack Overflow word, tag pairs).

Individual versus group representation of memory. Even with the large data sets used in this research, there was not enough data on each individual user to build up separate co-occurrence matrices for them. That is, the context component of the declarative memory models used in this research was formed as a collection of observed co-occurrences across all individuals. Nonetheless, the prior component for the models was customized for each individual (except in the Twitter popular hashtags dataset when it was not possible). Ideally, both components would be customized to the individual, and it remains an open question how much accuracy is influenced by using an aggregate or custom context component in the models.

Attentional weight. The Bayesian declarative memory model for ACT-R has an attentional weight term W_j that can be used to attenuate or remove stop words. The random permutation model was also easily modified to have an attentional weight term by attenuating each word's set of ones and negative ones values by that word's attentional weight. Both a method for stop-word removal (frequency filtering) and stop-word attenuating (entropy weighting) performed well for both models.

It is unclear at this point if this modification is cognitively plausible, even if the modification was cleanly added to the current ACT-R model. What can be said from this research is that using a proper method for handling the low-predictor words produced some of the largest improvements in model accuracy compared to the other explorations (e.g., compared with word order or properly adding terms). Consequently, it may be worthwhile to explore this area more thoroughly in the future.

Also, there may be other ways besides adjusting the attentional weight that these models can be modified to achieve the same performance improvement. The entropy weighting term is likely solving a problem that resides only in contextual activation and not prior activation, because the activation for the prior component for each word is independent of the presentation of other words, so stop words have no way to interact with tag prior activation in this case.

One alternative explanation is that the entropy weighting measure was needed because the contextual co-occurrence matrix was not yet stable, even with these large dataset sizes. If the matrix was not yet stable, then by chance certain stop words would contribute more net activation to specific tags and introduce noise into the retrieval process for that tag. So further weighting the contribution of these stop words by leveraging the attentional weight term ensured that this noise was suppressed in the system. If this is the case, it would be interesting to see how large a dataset is required to not need to modify the attentional weight term. However, even if this is the case, the random permutation model would still require entropy weighting for larger dataset sizes, since this model simply adds environment vectors to produce the current context vector and has no mechanism (unlike Bayesian normalization) for weighting the contribution of an environment vector.

Another explanation for the Bayesian model is that additional attentional weighting was needed because the contextual matrix was derived only from word by tag co-occurrences and not (word or tag) by (word or tag) co-occurrences. This was done for performance reasons as the computational complexity otherwise is simply too high for current hardware. It is possible that by adding all pairwise combinations of both words and tags in the posts may soften and stabilize the contribution of stop words when computing contextual activation for a tag, given current context.

Also perhaps a different S_{ji} model may not need entropy weighing at all (e.g., the Hebbian-inspired associative learning model described in Thomson & Lebiere, 2013). Further research could explore how entropy weighting interacts with other models of S_{ji}, and not just the random permutation and full Bayesian model.

Nonetheless, it seems likely that the entropy method for computing the attentional weight term W_j may be applicable to other declarative memory tasks. The entropy method is parameter-free (unlike the frequency-filtering method), completely data driven (much like the co-occurrence matrix), does not require tuning, and increases model accuracy for these data sets. It is common to handle stop words in some manner when working with NLP and large text corpuses. Therefore, if these large corpuses are used to test declarative memory, then the entropy method may be a more cognitively plausible way of dealing with stop words rather than filtering based on a predetermined list or the frequency count in the dataset.

Prior component. Adding a prior component to the random permutation model also increased model accuracy for both Stack Overflow and Twitter. It is unsurprising that this occurred. However, it is quite surprising that this is the first time that the prior component has been included and tested in a random permutation model. If the random permutation model is explored as a declarative memory theory for other data sets, it should be worthwhile to include the prior component for those explorations. Although it did not seem to matter much how the prior component is combined with the context component in this study, there may be better ways than the log-odds transformation to combine the terms. Further research could focus on exploring a larger set of methods for combining the context component of the random permutation model with the prior component from the ACT-R theory.

Activation calculation. The Bayesian and vector-based memory models differ in how the co-occurrence matrix is represented (lossy compression for vector-based models). They also differ in how the activation of each tag is calculated, given the different representations. The vector-based memory models, like LSA, compute a correlation between the context and each tag, while the Bayesian models sum the log likelihoods that each word in the context occurs with each tag. These two methods for computing the activation are correlated but not identical. Further research could try to tease apart how model accuracy is independently influenced by the representation (compressed or not) and the activation calculation (correlation or log likelihood). To do this, four models could be constructed (uncompressed and compressed representation crossed with correlation and log likelihood activation) compared to the two in this study, so that the effects of representation and activation calculation could be partialed out. This would allow, for example, the method of compression for the random permutation vector-based model to be analyzed independent of activation calculation. The results would help determine the specific representation that the declarative memory system uses to represent co-occurrences, and also the specific activation calculation that is carried out by the system.

Observational versus experimental design. All data used for this study were collected from publicly available data sets and were not subject to an experimental design (i.e., observational data). There are advantages and disadvantages to both observational and experimental designs. One primary advantage with the observational approach here is size and availability: Data for millions of posts and tweets are immediately available for analysis. However, one disadvantage is that there is no way to probe a user at a particular point in time, which could be quite advantageous for answering certain questions. One hybrid approach worth considering for future research is to use the publicly available data sets to build up a representation of declarative memory for each user. Afterward, bring a sample of these specific users into a laboratory setting and probe them with specific retrieval requests. Multiple users could be asked to tag the same message, and then models could be tested to see if they reliably predict differences in accuracy between the subjects.

Directly comparing the models. After including the prior and entropy components for both models, overall accuracy was similar for the Bayesian and random permutation models for this task. Therefore, it is unclear precisely which model is more cognitively plausible (or if both are correct and simply working at different levels of abstraction). However, there were two areas in particular where model performance showed relatively large differences: accuracy change as a function of corpus size and reliance on entropy weighting. These areas could be used in future research to more directly challenge the models. For example, a domain could be picked where enough data is collected to build a custom co-occurrence matrix for each user. Then one could look at model accuracy over time for each user as the co-occurrence matrix is increased to see if the trajectory is more accurately modeled by a random permutation (more accurate with lower *N*) or Bayesian (more accurate with higher *N*) model. Further, if a large-scale domain is picked and it is shown that the Bayesian model does not require entropy weighting for the co-occurrence matrix after the representations are fully stabilized, then this would provide a case where the two models are dramatically different: The random permutation model would require explicitly weighting the stop words (entropy weighting) while the Bayesian model would not.

Summary

This work has shown how large collections of publicly available behavioral data can be used to explore and test different theories of declarative memory and their individual architectural components. This in turn provided (a) a better understanding of how a user's past tagging history influences future tag use; (b) a more thorough evaluation of the strengths, weaknesses, and differences between two cognitively plausible memory models on large-scale real-world retrieval tasks; and (c) two efficient implementations of the models for making user-customized tag predictions for Stack Overflow posts and Twitter tweets.

This work has shown how past user history is important, and how a context-only model like the random permutation model can be modified to include the prior component. The work has also shown how the strength of the random permutation model may not be in its ability to represent word order, but rather in its ability to efficiently compress the full co-occurrence representation used for the Bayesian model. Several different formalisms for ACT-R's attentional weighting term were also explored, and an entropy-weighting technique was found to perform well without requiring any parameter tuning. Finally, this work has helped raise the question about how co-occurrence observations are represented at the neural level, because two models with quite different representations performed similarly well. The results from the study and the new research questions raised will help to better understand how imposing specific architectural constraints on memory models influence what information is retrieved.

References

Anderson, J. R. (1974). Retrieval of propositional information from long-term memory. *Cognitive Psychology, 6,* 451–474.

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York, NY: Oxford University Press.

Anderson, J. R., & Milson, R. (1989). Human memory: An adaptive perspective. *Psychological Review, 96,* 703.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python*. Sebastopol, CA: O'Reilly Media, Incorporated.

Budiu, R., Royer, C., & Pirolli, P. (2007). Modeling information scent: A comparison of LSA, PMI and GLSA similarity measures on common tests and corpora. In *Proceedings of the 8th Annual Conference of the Recherche d'Information Assistée par Ordinateur* (pp. 314–332). Pittsburgh, PA: Centre des Hautes Études Internationales d'Informatique Documentaire.

Cumming, G., & Finch, S. (2005). Inference by eye: Confidence intervals and how to read pictures of data. *American Psychologist, 60,* 170.

Denton, E., Weston, J., Paluri, M., Bourdev, L., & Fergus, R. (2015). User Conditional Hashtag Prediction for Images. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1731–1740). New York, NY: ACM.

Douglass, S. A., & Myers, C. W. (2010). Concurrent knowledge activation calculation in large declarative memories. In D. D. Salvucci & G. Gunzelmann (Eds.), *Proceedings of the 10th International Conference on Cognitive Modeling* (pp. 55–60). Philadelphia, PA: Drexel University.

Dumais, S. T. (1991). Improving the retrieval of information from external sources. *Behavior Research Methods, 23,* 229–236.

Efron, M. (2010). Hashtag retrieval in a microblogging environment. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 787–788). New York, NY: ACM.

Efron, M., & Golovchinsky, G. (2011). Estimation methods for ranking recent information. In *Proceedings of the 34th international ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 495–504). New York, NY: ACM.

Farahat, A., Pirolli, P., & Markova, P. (2004). *Incremental methods for computing word pair similarity* (tech. rep. no. TR-04–6-2004). Palo, Alto, CA: PARC, Inc.

Fu, W. T., & Pirolli, P. (2007). SNIF-ACT: A cognitive model of user navigation on the World Wide Web. *Human–Computer Interaction, 22,* 355–412.

Google. (2013). *Google I/O 2013: Keynote*. Retrieved from http://www.youtube.com/watch?v=9pmPa%7B_%7DKxsAM%7B#%7Dt=1h28m18s

Jones, M. N., & Mewhort, D. J. K. (2007). Representing word meaning and order information in a composite holographic lexicon. *Psychological Review, 114,* 1–37.

Kywe, S. M., Hoang, T. A., Lim, E. P., &Zhu, F. (2012, December). On recommending hashtags in twitter networks. In International Conference on Social Informatics (pp. 337–350). Springer Berlin Heidelberg.

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

Lewis, R. L., Vasishth, S., & Van Dyke, J. A. (2006). Computational principles of working memory in sentence comprehension. *Trends in Cognitive Sciences, 10,* 447–454.

Li, T., Wu, Y., & Zhang, Y. (2011). Twitter hash tag prediction algorithm. In *The 2011 International Conference on Internet Computing*. Las Vegas, NV: Citeseer.

Lin, Y.-R., Margolin, D., Keegan, B., Baronchelli, A., & Lazer, D. (2013). # Bigbirds Never Die: Understanding Social Dynamics of Emergent Hashtag. *arXiv preprint arXiv:1303.7144*.

Masson, M. E. J., & Loftus, G. R. (2003). Using confidence intervals for graphically based data interpretation. *Canadian Journal of Experimental Psychology, 57,* 203.

Mazzia, A., & Juett, J. (2009). *Suggesting hashtags on Twitter. EECS 545 (Machine Learning) Course Project Report*. Ann Arbor, MI: Department of Computer Science and Engineering, University of Michigan.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., & Smith, N. A. (2013). Improved part-of-speech tagging for online conversational

text with word clusters. In L. Vanderwende, H. Daumé, & K. Kirchhoff (Eds.), *Proceedings of NAACL-HLT* (pp. 380–390). Atlanta, GA: Association for Computational Linguistics.

Pirolli, P., & Fu, W.-T. (2003). SNIF-ACT: A model of information foraging on the World Wide Web. In F. Cena, A. Dattolo, E. W. De Luca, P. Lops, T. Plumbaum, & J. Vassileva (Eds.), *User Modeling 2003* (pp. 45–54). Johnstown, PA: Springer.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks, 6,* 623–641.

Recchia, G., Jones, M., Sahlgren, M., & Kanerva, P. (2010). Encoding sequential information in vector space models of semantics: Comparing holographic reduced representation and random permutation. In S. Ohlsson, & R. Catrambone (Eds.), *Proceedings of the 32nd Annual Conference of the Cognitive Science Society* (pp. 865–870). Austin, TX: Cognitive Science Society.

Rutledge-Taylor, M. F., & West, R. L. (2007). MALTA: Enhancing ACT-R with a holographic persistent knowledge store. In D. S. McNamara, & J. G. Trafton (Eds.), *Proceedings of the 24th Annual Conference of the Cognitive Science Society* (pp. 1433–1439). Austin, TX: Cognitive Science Society.

Rutledge-Taylor, M. F., & West, R. L. (2008). Modeling the fan effect using dynamically structured holographic memory. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 385–390). Austin, TX: Cognitive Science Society.

Sahlgren, M., Holst, A., & Kanerva, P. (2008). Permutations as a means to encode order in word space. In B. C. Love, K. McRae, & V. M. Sloutsky (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society* (pp. 1300–1305). Austin, TX: Cognitive Science Society.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing & management, 24,* 513–523.

StackExchange. (2014). *Creative commons data dump*. Retrieved from http://data.stackexchange.com/about

Stanley, C. (2014). Repository for dissertation. *GitHub*. Retrieved from https://github.com/claytontstanley/dissertation

Stanley, C., & Byrne, M. D. (2013). Predicting tags for Stack Overflow posts. In R. L. West, & T. C. Stewart (Eds.), *The 12th International Conference on Cognitive Modeling* (pp. 414–419). Ottawa, Canada: Carleton University.

Thomson, R., & Lebiere, C. (2013). Constraining Bayesian inference with cognitive architectures: An updated associative learning mechanism in ACT-R. In M. Knauff, M. Pauen, N. Sebanz, & I. Wachsmuth (Eds.), *Proceedings of the 35th Annual Conference of the Cognitive Science Society* (pp. 3539–3544).

Tryon, W. W. (2001). Evaluating statistical difference, equivalence, and indeterminacy using inferential confidence intervals: An integrated alternative method of conducting null hypothesis statistical tests. *Psychological methods, 6,* 371.

Yang, L., Sun, T., Zhang, M., & Mei, Q. (2012). We Know What @You #Tag: Does the dual role affect hashtag adoption? In *Proceedings of the 21st international conference on World Wide Web* (pp. 261–270). New York, NY: ACM.

Zangerle, E., Gassler, W., & Specht, G. (2011). Recommending #-tags in Twitter. In F. Cena, A. Dattolo, E. W. De Luca, P. Lops, T. Plumbaum, & J. Vassileva (Eds.), *Proceedings of the Workshop on Semantic Adaptive Social Web* (pp. 67–78).

Received May 28, 2015
Revision received June 16, 2016
Accepted June 19, 2016 ■