Dynamic Word Embeddings for Evolving Semantic Discovery

Zijun Yao
Rutgers University
zijun.yao@rutgers.edu

Yifan Sun
Technicolor Research
yifan.sun@technicolor.com

Weicong Ding
Amazon
20008005dwc@gmail.com

Nikhil Rao
Amazon
nikhilrao86@gmail.com

Hui Xiong
Rutgers University
hxiong@rutgers.edu

ABSTRACT

Word evolution refers to the changing meanings and associations of words throughout time, as a byproduct of human language evolution. By studying word evolution, we can infer social trends and language constructs over different periods of human history. However, traditional techniques such as word representation learning do not adequately capture the evolving language structure and vocabulary. In this paper, we develop a dynamic statistical model to learn time-aware word vector representation. We propose a model that simultaneously learns time-aware embeddings and solves the resulting "alignment problem". This model is trained on a crawled NYTimes dataset. Additionally, we develop multiple intuitive evaluation strategies of temporal word embeddings. Our qualitative and quantitative tests indicate that our method not only reliably captures this evolution over time, but also consistently outperforms state-of-the-art temporal embedding approaches on both semantic accuracy and alignment quality.

ACM Reference Format:
Zijun Yao, Yifan Sun, Weicong Ding, Nikhil Rao, and Hui Xiong. 2018. Dynamic Word Embeddings for Evolving Semantic Discovery. In *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159703

1 INTRODUCTION

Human language is an evolving construct, with word semantic associations changing over time. For example, `apple` which was traditionally only associated with fruits, is now also associated with a technology company. Similarly, the association of names of famous personalities (*e.g.*, `trump`) changes with a change in their roles. For this reason, understanding and tracking word evolution is useful for time-aware knowledge extraction tasks (*e.g.*, public sentiment analysis), and other applications in text mining. To this end, we aim to learn word embeddings with a temporal bent, for capturing time-aware meanings of words.

Word embeddings aim to represent words with low-dimensional vectors, where words with similar semantics are geometrically closer (*e.g.* `red` and `blue` are closer than `red` and `squirrel`). Classic word embedding techniques started in the 90s and relied on statistical approaches [9, 20]. Later, neural network approaches [5], as well as recent advances such as word2vec [24, 25] and GloVE [27] have greatly improved the performance of word representation learning. However, these techniques usually do not consider temporal factors, and assume that the word is static across time.

In this paper, we are interested in computing time-aware embedding of words. Specifically, each word in a different time frame (*e.g.*, years) is represented by a different vector. From these embeddings, we have a better notion of "distance" (the cosine distance between word embedding vectors), and by looking at word "neighborhoods" (defined through this distance), we can better understand word associations as well as word meanings, as they evolve over time.

For instance, by locating the embeddings of personality names such as `trump`'s closest words, we can see that he can be associated with the trajectory : `real estate` → `television` → `republican`. Similarly, the trajectory of `apple` travels from the neighborhood of `strawberry, mango` to that of `iphone, ipad`.

A key practical issue of learning different word embeddings for different time periods is alignment. Specifically, most cost functions for training are invariant to rotations, as a byproduct, the learned embeddings across time may not be placed in the same latent space. We call this the *alignment* problem, which is an issue in general if embeddings are learned independently for each time slice.

Unlike traditional methods, literature on learning temporal word embedding is relatively short: [12, 15, 19, 40]. In general, the approaches in these works follow a similar two-step pattern: first compute static word embeddings in each time slice separately, then find a way to align the word embeddings across time slices. To achieve alignment, [15] finds a linear transformation of words between any two time slices by solving a d-dimensional least squares problem of k nearest neighbor words (where d is the embedding dimension). Additionally, [40] also use the linear transformation approach between a base and target time slices, and computes the linear transformation using anchor words, which does not change meaning between the two time slices. This method requires the prior knowledge of words that are in fact static, which involves additional expert supervision. Finally, [12] imposes the transformation to be orthogonal, and solves a d-dimensional Procrustes problem between every two adjacent time slices.

Our main novelty in this paper is to learn the word embeddings across time jointly, thus obviating the need to solve a separate alignment problem. Specifically, we propose to learn temporal embeddings in all time slices concurrently, and apply regularization

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159703

terms to smooth embedding changes across time. There are three main advantages to this proposed joint modeling approach:

- First, this can be seen as an improvement over traditional, "single-time" methods such as word2vec.
- Second, our experimental results suggest that enforcing alignment through regularization yields better results than two-step methods.
- Third, we can share information across time slices for the majority of vocabulary. As a result, our method is robust against data sparsity – we can afford to have time slices where some words are rarely present, or even missing. This is a crucial practical advantage offered by our method.

Since our model requires embeddings across all time slices to be learned at the same time, it can be computationally challenging. To mitigate this, we employ a block coordinate descent method which can handle large vocabulary sizes through decomposition.

In experimental study, we learn temporal embeddings of words from The New York Times articles between 1990 and 2016. In contrast, previous temporal word embedding works have focused on time-stamped novels and magazine collections (such as Google N-Gram and COHA). However, news corpora are naturally advantageous to studying language evolution through the lens of current events. In addition, it allows us to work with a corpus that maintains consistency in narrative style and grammar, as opposed to Facebook and Twitter posts. For evaluating our embeddings, we develop both qualitative and quantitative metrics.

Qualitatively, we illustrate the advantages of temporal embeddings for evolving semantics discovery by 1) plotting word vector trajectories to find evolving meanings and associations, 2) using alignment through time to identify associated words across time, and 3) looking at norms as a representative of concept popularity.

Quantitatively, we first use semantic topics extracted from Sections (*e.g.*, *Technology*, *World News*) of news articles as ground truth to evaluate the semantic accuracy of temporal embeddings. Additionally, we provide two testsets to evaluate cross-time alignment quality: one consists of known changing roles (*e.g.*, U.S. presidents), determined objectively, and one of concept replacements (*e.g.*, compact disk to mp3), determined more subjectively. The experimental results show the effectiveness of our proposed model and demonstrate substantial improvements against baseline methods.

The rest of this paper is organized as follows. In Section 2, we present the proposed model for temporal embedding learning, and in Section 3, we describe a scalable algorithm to train it. In Section 4, we describe the news corpus dataset and setup details of experiments. We perform qualitative evaluations in Section 5. Finally, we quantitatively compare our embeddings against other state-of-the-art temporal embeddings in Section 6.

2 METHODOLOGY

We now set up our temporal word embedding model. We consider a text corpus collected across time. These kinds of corpora such as news article collections with published dates or social media discussion with time-stamps are ubiquitous. Formally, we denote by $\mathcal{D} = (\mathcal{D}_1, \dots, \mathcal{D}_T)$ our text corpus where each $\mathcal{D}_t$, $t = 1, \dots T$, is the corpus of all documents in the t-th time slice. Without loss of generality, we assume the time slices are ordered chronologically. The length of these time slices can be on the order of months, years, or decades. Moreover, the length of all the time slices could be different. We consider an overall vocabulary $\mathcal{V} = \{w_1, \dots, w_V\}$ of size V. We note that the vocabulary $\mathcal{V}$ consists of words present in the corpus at any point in time, and thus it is possible for some $w \in \mathcal{V}$ to not appear at all in some $\mathcal{D}_t$. This includes emerging words and dying words that are typical in real-world news corpora.

Given such a time-tagged corpus, our goal is to find a dense, low-dimensional vector representation $u_w(t) \in \mathbb{R}^d$, $d \ll V$ for each word $w \in \mathcal{V}$ and each time period $t = 1, \dots, T$. We denote by u_w the static embedding for word w (for example, learned via word2vec), and d is the embedding dimension (typically $50 \le d \le 200$). Compactly, we denote by $U(t)$ (of size $V \times d$) the embedding matrix of all words whose i-th row corresponds to the embedding vector of i-th word $u_{w_i}(t)$.

2.1 Time-agnostic word embeddings

A fundamental observation in static word embedding literature is that semantically similar words often have similar neighboring words in a corpus [10]. This is the idea behind learning dense low-dimensional word representations both traditionally [5, 9, 20] and recently [24, 27]. In several of these methods, the neighboring structure is captured by the frequencies by which pairs of words co-occur within a small local window.

We compute the $V \times V$ pointwise mutual information (PMI) matrix specific to a corpus $\mathcal{D}$, whose w, c-th entry is:

$$\mathrm{PMI}(\mathcal{D}, L)_{w,c} = \log\left(\frac{\#(w,c) \cdot |\mathcal{D}|}{\#(w) \cdot \#(c)}\right), \tag{1}$$

where $\#(w,c)$ counts the number of times that words w and c co-occur within a window of size L in corpus $\mathcal{D}$, and $\#(w)$, $\#(c)$ counts the number of occurrences of words w and c in $\mathcal{D}$. $|\mathcal{D}|$ is total number of word tokens in the corpus. L is typically around 5 to 10; we set $L = 5$ throughout this paper.

The key idea behind both word2vec [24] and GloVE [27] is to find embedding vectors u_w and u_c such that for any w, c combination,

$$u_w^T u_c \approx \mathrm{PMI}(\mathcal{D}, L)_{w,c}, \tag{2}$$

where each u_w has length $d \ll V$. While both [24] and [27] offer highly scalable algorithms such as negative sampling to do this implicitly, [17] shows that these are equivalent to low-rank factorization of $\mathrm{PMI}(\mathcal{D}, L)$ [1]. Our approach is primarily motivated by this observation. We note that though the PMI matrices are of size $V \times V$, in real-world datasets it is typically sparse as observed in [27], for which efficient factorization methods exist [39].

2.2 Temporal word embeddings

A natural extension of the static word embedding intuition is to use this matrix factorization technique on each time slice $\mathcal{D}_t$ separately. Specifically, for each time slice t, we define the w, c-th entry of *positive* PMI matrix ($\mathrm{PPMI}(t, L)$) as [2]

$$\mathrm{PPMI}(t, L)_{w,c} = \max\{\mathrm{PMI}(\mathcal{D}_t, L)_{w,c}, 0\}. := Y(t). \tag{3}$$

[1]with a constant shift that can be zero.

[2]We consider the PPMI rather than the PMI because when $\frac{\#(w,c)\cdot|\mathcal{D}|}{\#(w)\cdot\#(c)}$ is very small, taking the log results in large negative values and is thus extremely unstable. Since for most significantly related pairs w and c the log argument is > 1, thresholding it in this way will not affect the solution significantly, but will offer much better numerical stability. This approach is not unique to us; [17] also factorizes the PPMI.

The temporal word embeddings $U(t)$ must satisfy

$$U(t)U(t)^T \approx \text{PPMI}(t, L). \tag{4}$$

One way to find such $U(t)$ is for each t, factorizing PPMI(t, L) by either using an eigenvalue method or solving a matrix factorization problem iteratively.

The Alignment Problem: Imposing (4) is not sufficient for a unique embedding, since the solutions are invariant under rotation; that is, for any $d \times d$ orthogonal matrix R, we have the embedding $\widehat{U}(t) = U(t)R$, the approximation error in (4) is the same since

$$\widehat{U}(t)\widehat{U}(t)^T = U(t)RR^TU(t)^T = U(t)U(t)^T.$$

For this reason, it is important to enforce *alignment*; if word w did not semantically shift from t to $t+1$, then we additionally require $u_w(t) \approx u_w(t+1)$.

To do this, [12, 15] propose two-step procedures; first, they factorize each $Y(t)$ separately, and afterwards enforce alignment using local linear mapping [15] or solving an orthogonal procrustes problem [12]. Note that in these methods, aligning $U(t)$ to $U(t')$ assumes that we desire $U(t) \approx U(t')$. If we only pick $t' = t+1$ (as done in [12]), this assumption is reasonable because between any two years, only a few words experience semantic shift, emergence, or death. However, this becomes problematic if $U(t)$ was a result of a time period with extremely sparse data (and hence poorly learned); all subsequent year embeddings and previous year embeddings will be poorly aligned.

2.3 Our model

We propose finding temporal word embeddings as the solution of the following joint optimization problem:

$$\begin{aligned}\min_{U(1),\ldots,U(T)} \frac{1}{2}\sum_{t=1}^{T} \|Y(t) - U(t)U(t)^T\|_F^2 \\ + \frac{\lambda}{2}\sum_{t=1}^{T}\|U(t)\|_F^2 + \frac{\tau}{2}\sum_{t=2}^{T}\|U(t-1) - U(t)\|_F^2,\end{aligned} \tag{5}$$

where $Y(t) = \text{PPMI}(t, L)$ and $\lambda, \tau > 0$. Here the penalty term $\|U(t)\|_F^2$ enforces the low-rank data-fidelity as widely adopted in previous literature. The key smoothing term $\|U(t-1) - U(t)\|_F^2$ encourages the word embeddings to be aligned. The parameter τ controls how fast we allow the embeddings to change; $\tau = 0$ enforces no alignment, and picking $\tau \to \infty$ converges to a static embedding with $U(1) = U(2) = \ldots = U(T)$. Note that the methods of [12, 15] can be viewed as suboptimal solutions of (5), in that they optimize for each term separately. For one, while the strategies in [15] and [12] enforce alignment pairwise, we enforce alignment across *all* time slices; that is, the final aligned solution $U(t)$ is influenced by not only $U(t-1)$ and $U(t+1)$, but every other embedding as well. This avoids the propagation of alignment errors caused by a specific time frame's subsampling. Additionally, consider an extreme case in which word w is absent from $\mathcal{D}_t$ but has similar meaning in both $t-1$ and $t+1$. Directly applying any matrix factorization technique to each time point would enforce $u_w(t) \approx \mathbf{0}$. However, for the right choice of τ, the solution $u_w(t)$ to (5) will be close to $u_w(t-1)$ and $u_w(t+1)$. Overall, our method achieves high fidelity embeddings with a much smaller corpus, and in particular, in Section 6, we demonstrate that our embeddings are robust against sudden undersampling of specific time slices.

3 OPTIMIZATION

A key challenge in solving (5) is that for large V and T, one might not be able to fit all the PPMI matrices $Y(1), \ldots, Y(T)$ in memory, even though $Y(t)$ is sparse. Therefore, for scalability, an obvious solution is to first decompose the objective across time, using alternating minimization to solve for $U(t)$ at each step:

$$\begin{aligned}\min_{U(t)} \overbrace{\frac{1}{2}\|Y(t) - U(t)U(t)^T\|_F^2 + \frac{\lambda}{2}\|U(t)\|_F^2}^{f(U(t))} \\ + \frac{\tau}{2}\left(\|U(t-1) - U(t)\|_F^2 + \|U(t) - U(t+1)\|_F^2\right)\end{aligned} \tag{6}$$

for a specific t. Note that $f(U(t))$ is quartic in $U(t)$, and thus even if we only solve for a fixed t, (6) cannot be minimized analytically. Thus, our only option is to solve (6) iteratively using a fast first-order method such as gradient descent. The gradient of the first term alone is given by

$$\nabla f(U(t)) = -2Y(t)U(t) + 2U(t)U(t)^TU(t). \tag{7}$$

Each gradient computation is of the order $O(nnz(Y(t))d + d^2V)$ (which is then nested in iteratively optimizing $U(t)$ for each t).[3] In practical applications, V is in the order of ten-thousands to hundred-thousands, and T is in the order of tens to hundreds.

Let us instead look at a slightly relaxed problem of minimizing

$$\begin{aligned}\min_{U(t),W(t)} \frac{1}{2}\sum_{t=1}^{T}\|Y(t) - U(t)W(t)^T\|_F^2 + \frac{\gamma}{2}\sum_{t=1}^{T}\|U(t) - W(t)\|_F^2 \\ + \frac{\lambda}{2}\sum_{t=1}^{T}\|U(t)\|_F^2 + \frac{\tau}{2}\sum_{t=2}^{T}\|U(t-1) - U(t)\|_F^2 \\ + \frac{\lambda}{2}\sum_{t=1}^{T}\|W(t)\|_F^2 + \frac{\tau}{2}\sum_{t=2}^{T}\|W(t-1) - W(t)\|_F^2,\end{aligned} \tag{8}$$

where variables $W(t), t = 1, \ldots, T$ are introduced to break the symmetry of factorizing $Y(t)$. Now, minimizing for each $U(t)$ (and equivalently $W(t)$) is just the solution of a ridge regression problem, and can be solved in one step by setting the gradient of the objective of (8) to 0, *i.e.* $U(t)A = B$ where

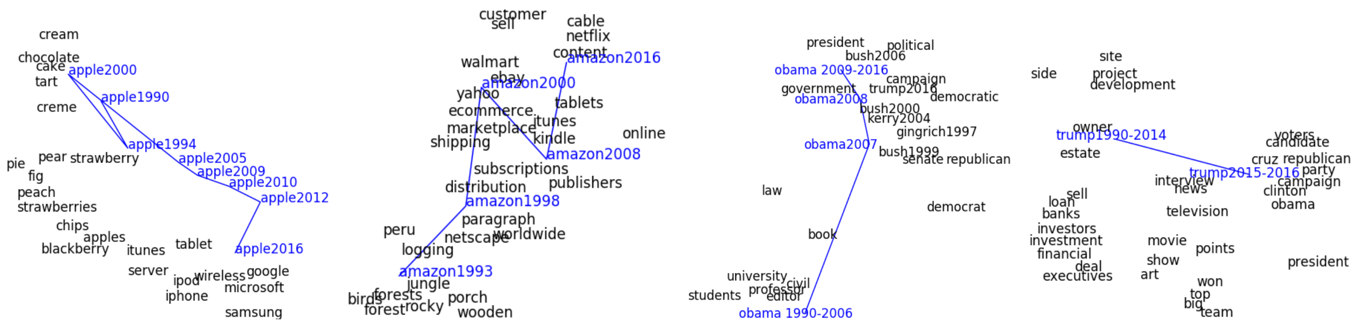
$$A = W(t)^TW(t) + (\gamma + \lambda + 2\tau)I,$$

$$B = Y(t)W(t) + \gamma W(t) + \tau(U(t-1) + U(t+1))$$

for $t = 2, \ldots, T-1$, and constants adjusted for $t = 0, T$. Forming A and B requires $O(Vd^2 + nnz(Y(t))d)$, and solving $U(t)A = B$ can be done in $O(d^3)$ computations, in one step. This can be further decomposed to row-by-row blocks of size b, by minimizing over a row block of $U(t)$ at a time, which reduces the complexity of forming A and B to $O(bd^2 + nnz(Y(t)[:b, :])d)$, *i.e.*independent of V. This allows scaling for very large V, as only one row block of $Y(t)$ must be loaded at a time.

Block coordinate descent vs. stochastic gradient descent: The method described here is commonly referred to as *block coordinate descent (BCD)* because it minimizes with respect to a single block ($U(t)$ or $W(t)$) at a time, and the block size can be made even smaller (a few rows of $U(t)$ or $W(t)$) to maintain scalability. The main appeal of BCD is scalability [39]; however, a main drawback is lack of convergence guarantees, even in the case of convex optimization [29].

[3] $nnz(\cdot)$ is the number of nonzeros in the matrix.



(a) apple **(b) amazon** **(c) obama** **(d) trump**

Figure 1: Trajectories of brand names and people through time: `apple`, `amazon`, `obama`, and `trump`.

In practice, however, BCD is highly successful and has been used in many applications [38]. Another choice of optimization is stochastic gradient descent (SGD), which decomposes the *objective* as a sum of smaller terms. For example, the first term of (8) can be written as a sum of terms, each using only one row of $Y(t)$:

$$f(U(t)) = \sum_{i=1}^{V} \|Y(t)[i,:] - u_i(t)^T W(t)\|_F^2. \quad (9)$$

The complexity at first glance is smaller than that of BCD; however, SGD comes with the well-documented issues of slow progress and hard-to-tune step sizes, and in practice, can be much slower for matrix factorization applications [30, 39]. However, we point out that the choice of the optimization method is agnostic to our model; anything that successfully solves (5) should lead to an equally successful embedding.

4 EXPERIMENTAL DATASET AND SETUP

In this section we describe the specific procedure used to generate embeddings for the next two sections.

News article dataset: First, we crawl a total of 99,872 articles from the New York Times, published between January 1990 and July 2016.[4] In addition to the text, we also collected metadata including title, author, release date, and section label (*e.g.*, Business, Sports, Technology); in total, there are 59 such sections. We use yearly time slices, dividing the corpus into $T = 27$ partitions. After removing rare words (fewer than 200 occurrences in all articles across time) and stop words, our vocabulary consists of $V = 20,936$ unique words. We then compute a co-occurrence matrix for each time slice t with a window size $L = 5$, which is then used to compute the PPMI matrix as outlined in (3). All the embedding methods that we compared against are trained on this same dataset.

Training details for our algorithm: We perform a grid search to find the best regularization and optimization parameters. As a result of our search, we obtain $\lambda = 10$, $\tau = \gamma = 50$, and run for 5 epochs (5 complete pass over all time slices, and all rows and columns of $Y(t)$). Interestingly, setting $\lambda = 0$ also yielded good results, but required more iterations to converge. The block variable is one matrix ($U(t)$ or $V(t)$ for a specific t).

Distance metric: All distances between two words are calculated by the cosine similarity between embedding vectors:

[4] The data is available at: https://sites.google.com/site/zijunyaorutgers/.

$$\text{similarity}(a, b) = \text{cosine}(u_a, u_b) = \frac{u_a^T u_b}{\|u_a\|_2 \cdot \|u_b\|_2}, \quad (10)$$

where u_a and u_b are the embeddings of words a and b.

5 QUALITATIVE EVALUATION

The embeddings we learn reveal interesting patterns in the shift of word semantics, cross-time semantic analogy, and popularity trends of concepts from the news corpus.

5.1 Trajectory visualization

The trajectory of a word in the (properly aligned) embedded space provides tools to understand the shift in meanings of words over time. This can help broader applications, such as capturing and quantifying linguistic evolution, characterizing brands and people, and analyzing emerging association between certain words.

Figure 1 shows the trajectories of a set of example words. We plot the 2-D t-SNE projection of each word's temporal embedding across time. We also plot the closest words to the target word from each time slice. We pick four words of interest: `apple` and `amazon` as emerging corporate names while originally referring to a fruit and a rainforest, and `obama` and `trump` as people with changing professional roles.

In all cases, the embeddings illustrate significant semantic shifts of the words of interest during this 27-year time frame. We see `apple` shift from a fruit and dessert ingredient to space of technology. Interestingly, there is a spike in 1994 in the trajectory, when Apple led a short tide of discussion because of the replacement of the CEO and a collaboration with IBM; then the association shifted back to neighborhood of fruit and dessert until the recovery by Steve Jobs in early 2000s. Similarly, `amazon` shifts from a forest to an e-commerce company, finally landing in 2016 as a content creation and online-streaming provider due to the popularity of its Prime Video service. The US president names, `obama` and `trump`, are most telling, shifting from their pre-presidential lives (Obama as a civil rights attorney and university professor; Trump as a real estate developer and TV celebrity) to the political sphere. Overall, Figure 1 demonstrates that first, our temporal word embeddings can well capture the semantic shifts of words across time, and second, our model provides high alignment quality in that same-meaning words across different years have geometrically close embeddings, without having to solve a separate optimization problem for alignment.

Table 1: Equivalent technologies through time: `iphone`, `twitter`, and `mp3`.

Query	`iphone`, 2012	`twitter`, 2012	`mp3`, 2000
90-94	desktop, pc, dos, macintosh, software	broadcast, cnn, bulletin, tv, radio, messages, correspondents	stereo, disk, disks, audio
95-96			mp3
97		chat, messages, emails, web	
98-02	pc		
03			napster
04			mp3
05-06	ipod	blog, posted	itunes, downloaded
07-08	iphone		
09-12		twitter	
13-16	smartphone, iphone		

5.2 Equivalence searching

Another key advantage of word alignment is the ability to find conceptually "equivalent" items or people over time. We provide examples in the field of technology, official roles, and sports professionals. In this type of test, we create a query consisting of a word-year pair that is particularly the representative of that word in that year, and look for other word-year pairs in its vicinity, for different years.

Table 1 lists the closest words (*top*-1) of each year to the query vector. For visualization purpose we lump semantically similar words together. For example, the first column shows that `iphone` in 2012 is closely associated with smartphones in recent years, but is close to words such as `desktop` and `macintosh` in the 90's; interestingly, `telephone` never appears, suggesting the iPhone serves people more as a portable computer than a calling device. As another example, by looking at the trajectory of `twitter`, we see the evolution of news sources, from TV & radio news broadcasts in the 90s to chatrooms, websites, and emails in the early 2000s, blogs in the late 2000s, and finally tweets today. The last example is fairly obvious; `mp3` represents the main form of which music is consumed in 2000, replacing disk and stereo in 1990s (`cassette` also appears in *top*-3) and is later replaced by online streaming. We can see a one-year spike of Napster which was shut down because of copyright infringement[5], and later a new streaming service - iTunes.

Next, we use embeddings to identify people in political roles. Table 2 attempts to discover who is the U.S. president[6] and New York City mayor[7] of the time, using as query `obama` in 2016 and `blasio` in 2015. For president, only the closest word from each year is listed, and is always correct (accounting for the election years). For mayor, the top-1 closet word is shown unless it is `mayor`, in which case the second word is shown. We can see that both the roles of US president and NYC mayor have been well searched for different

[5]Napster ended its streaming service in 2001, so our equivalence is captured 2 years late; this delay could be because though the event happened in 2001, the legal ramifications were analyzed heavily in subsequent years.
[6]All data was scraped about half a year before Donald Trump was elected as U.S. president in 2016.
[7]We intentionally choose New York City because it is the most heavily discussed city in the New York Times.

Table 2: "Who governed?" The closest word to `obama` at year 2016 (role as president of United State) and `blasio` at year 2015 (role as mayor of New York City (NYC)). The stars indicate incorrect answers.

Question	US president	NYC mayor
Query	`obama`, 2016	`blasio`, 2015
90-92	bush	dinkins
93	clinton	
94-00		giuliani
01	bush	
02-05		bloomberg
06		n/a*
07		bloomberg
08		
09-10	obama	
11		cuomo*
12		bloomberg
13-16		blasio

Table 3: "Who was the ATP No.1 ranked male player?" The closest word to `nadal` at year 2010 for each year is listed. The correct answer is based on ATP year-end ranking and are bolded in the table.

year	1990	1991	1992	1993
word	**edberg**	lendl	sampras	**sampras**
1994	1995	1996	1997	1998
sampras	**sampras**	ivanisevic	**sampras**	**sampras**
1999	2000	2001	2002	2003
sampras	sampras	agassi	capriati	**roddick**
2004	2005	2006	2007	2008
federer	**federer**	roddick	**federer**	**nadal**
2009	2010	2011	2012	2013
federer	**nadal**	**djokovic**	federer	federer
2014	2015			
federer	**djokovic**			

persons in their terms of service. We see that the embedding for the President is consistent, and for the most part, so is that of the mayor of NYC. In 2011, `cuomo` is also partially relevant since he was the governor of NY state. We did not find any relevant words in query NYC mayor in year 2006.

Finally, we search for equivalences in sports, repeating the experiment for the ATP rank 1 male tennis player as shown in Table 3. In the case of president and mayor, we are heavily assisted by the fact that they are commonly referred to by a title: "President Obama" and "Mayor de Blasio". Tennis champions, on the other hand, are not referred by titles. Still, a surprising number of correct champions appear as the closest words, and all the names are those of famous tennis players for the given time period. A more exhaustive empirical study of alignment quality is provided in Section 6.

5.3 Popularity determination

It has often been observed that word embeddings computed by factorizing PMI matrices have norms that grow with word frequency [2, 27]. These word vector norms across time can be viewed as a

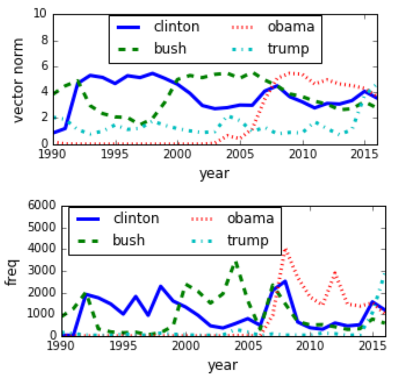


Figure 2: Norm (top) and relative frequency (bottom) throughout years 1990-2016. We select the names of U.S presidents within this time frame - `clinton`, `bush`, `obama`, and `trump`. We note that `bush` could refer to George H. W. Bush (1989-1993) or George W. Bush (2001-2009). Clinton could refer to Bill Clinton (1993-2001) or Hillary Clinton (U.S. secretary of state in 2009-2013 and U.S. presidential candidate in 2014-2016).

time series for detecting the trending concepts (*e.g.*, sudden semantic shifts or emergences) behind words, with more robustness than word frequency.

Figures 2 and 3 illustrate the comparison between embedding norm and frequency for determining *concept* popularity per year, determined by key words in the New York Times corpus. Generally, comparing to frequencies which are much more sporadic and noisy, we note that the norm of our embeddings encourages smoothness and normalization while being indicative of the periods when the corresponding words were making news rounds. In Figure 2, the embedding norms display nearly even 4-year humps corresponding to each president's term. In every term, the name of each current president becomes a trending concept which plays an important role in the information structure at the time. Two interesting observations can be gleaned. First, since Hillary Clinton continuously served as Secretary of State during 2009-2013, the popularity of `clinton` was preserved; however it was still not as popular as president `obama`. Second, because of the presidential campaign, `trump` in 2016 has a rising popularity that greatly surpasses that of his former role as a business man, and eventually surpasses his opponent `clinton` in terms of news coverage.

In Figure 3, we can see smooth rises and falls of temporary phenomena (the `enron` scandal and `qaeda` rises). For `qaeda`, we see that there is a jump in 2001, and then it remains steady with a small decline. In contrast, `enron` shows a sharper decline, as despite its temporal hype, it did not linger in the news long. Note also the stability of using norms to track popularity over frequency, which spikes for `enron` above `qaeda`, although 9/11 was far more news-prevalent than the corporation's scandalous decline. For the basketball star `pippen`, although his publicity (*e.g.*, frequency) was relatively fewer than business terms, his popularity is still recognized by the enhancement in vector norm. For another term `isis`, we can see that it begins to replace `qaeda` as the "trending terrorist organization" in news media.

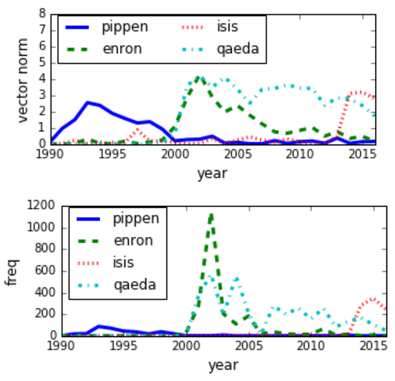


Figure 3: Norm (top) and relative frequency (bottom) per year in corpus of major event keywords: `pippen` for basketball stardom, `enron` for corporation scandal, `qaeda` and `isis` for emerging terrorism groups.

6 QUANTITATIVE EVALUATION

In this section, we empirically evaluate our proposed Dynamic Word2Vec model (DW2V) against other temporal word embedding methods.[8] In all cases, we set the embedding dimension to $d = 50$. We have the following baselines:

- **Static-Word2Vec (SW2V)**: the standard word2vec embeddings [25], trained on the entire corpus and ignoring time information.
- **Transformed-Word2Vec (TW2V) [15]**: the embeddings $U(t)$ are first trained separately by factorizing PPMI matrix for each year t, and then transformed by optimizing a linear transformation matrix which minimizes the distance between $u_w(t)$ and $u_w(t')$ for the $k = 30$ nearest words' embeddings to the querying word w.
- **Aligned-Word2Vec (AW2V) [12]**: the embeddings $U(t)$ are first trained by factorizing the PPMI matrix for each year t, and then aligned by searching for the best othornormal transformation between $U(t)$ and $U(t+1)$.

6.1 Semantic similarity

One of the most important properties of a word embedding is how accurately it carries the meaning of words. Therefore, we develop a test to see if words can be categorized by meaning based on embeddings. The news articles we collected are tagged with their "sections" such as *Business*, *Sports*. This information can be used to determine temporal word meanings. It is important to note that this information is not used in the word embedding learning. For example, we see that `amazon` occurs 41% of the time in *World* in 1995, associating strongly with the rainforest (not part of the USA), and 50% of the time in *Technology* in 2012, associating strongly with e-commerce. We thus use this to establish a ground truth of word category, by identifying words in years that are exceptionally numerous in one particular news section. That is, if a word is extremely frequent in a particular section, we associate that word with that section and use that as ground truth. We select the 11

[8]The testsets are available at: https://sites.google.com/site/zijunyaorutgers/.

Table 4: Normailized Mutual Information (NMI).

Method	10 Clusters	15 Clusters	20 Clusters
SW2V	0.6736	0.6867	0.6713
TW2V	0.5175	0.5221	0.5130
AW2V	0.6580	0.6618	0.6386
DW2V	**0.7175**	**0.7162**	**0.6906**

Table 5: F-measure (F_β).

Method	10 Clusters	15 Clusters	20 Clusters
SW2V	0.6163	0.7147	0.7214
TW2V	0.4584	0.5072	0.5373
AW2V	0.6530	0.7115	0.7187
DW2V	**0.6949**	**0.7515**	**0.7585**

most popular and discriminative sections[9] of the New York Times, and for each section s and each word w in year t, we compute its percentage p of occurrences in each section. To avoid duplicated word-time-section $< w, t, s >$ triplets, for a particular w and s we only keep the year of the largest strength, and additionally filter away any triplet with strength less than $p = 35\%$. Note that a random uniform distribution of words would result in it appearing about 9% of the time in each section, and our threshold is about 4 times that quantity. We do this to say with sufficient confidence that such associations can be treated as ground truth.

To limit the size differences among categories, for every section s with more than 200 qualified triplets, we keep the *top*-200 words by strength. In total, this results in 1888 triplets across 11 sections, where every word-year pair is strongly associated with a section as its true category.

We then apply spherical k-means, which uses cosine similarity between embeddings as the distance function for clustering, with $K = 10$, 15, and 20 clusters. We use two metrics to evaluate the clustering results:

- *Normalized Mutual Information (NMI)*, defined as
$$NMI(L, C) = \frac{I(L; C)}{[H(L) + H(C)]/2}, \quad (11)$$
where L represents the set of labels and C the set of clusters. $I(L; C)$ denotes the sum of mutual information between any cluster c_i and any label l_j, and $H(L)$ and $H(C)$ the entropy for labels and clusters, respectively. This metric evaluates the purity of clustering results from an information-theoretic perspective.
- *F_β-measure ($\mathbf{F}_\beta$)*, defined as
$$F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}, \quad (12)$$
where $P = \frac{TP}{TP+FP}$ denotes the precision and $R = \frac{TP}{TP+FN}$ denotes the recall. (TP/FP = true/false positive, TN/FN = true/false negative.) As an alternative method to evaluate clustering, we can view every pair of words as a series of decisions. Pick any two (w, t) pairs. If they are clustered together and additionally have the same section label, this is a correct decision; otherwise, the clustering performed a wrong decision. The metric F_β measures accuracy as the (β-weighted) harmonic mean of the precision and recall. We set $\beta = 5$ to give more weight to recall by penalizing false negative more strongly.

[9] Arts, Business, Fashion & Style, Health, Home & Garden, Real Estate, Science, Sports, Technology, U.S., World.

Table 6: Mean Reciprocal Rank (MRR) and Mean Precision (MP) for Testset 1.

Method	MRR	MP@1	MP@3	MP@5	MP@10
SW2V	0.3560	0.2664	0.4210	0.4774	0.5612
TW2V	0.0920	0.0500	0.1168	0.1482	0.1910
AW2V	0.1582	0.1066	0.1814	0.2241	0.2953
DW2V	**0.4222**	**0.3306**	**0.4854**	**0.5488**	**0.6191**

Table 7: Mean Reciprocal Rank (MRR) and Mean Precision (MP) for Testset 2.

Method	MRR	MP@1	MP@3	MP@5	MP@10
SW2V	0.0472	0.0000	0.0787	0.0787	0.2022
TW2V	0.0664	0.0404	0.0764	0.0989	0.1438
AW2V	0.0500	0.0225	0.0517	0.0787	0.1416
DW2V	**0.1444**	**0.0764**	**0.1596**	**0.2202**	**0.3820**

Tables 4 and 5 show the clustering evaluation. We can see that our proposed DW2V consistently outperforms other baselines for all values of K. These results show two advantages. First, the word semantic shift has been captured by the temporal embeddings (for example, by correlating correctly with the section label of `amazon`, which changes from *World* to *Technology*). Second, since embeddings of words of all years are used for clustering, a good clustering result indicates good alignment across years. We can also see that AW2V also performs well, as it also applies alignment between adjacent time slices for all words. However, TW2V does not perform well as others, suggesting that aligning locally (only a few words) is not sufficient for high alignment quality.

6.2 Alignment quality

We now more directly evaluate alignment quality, *i.e.* the property that the semantic distribution in temporal embedding space should be consistent over time. For example, if a word such as `estate` or `republican` does not change much throughout time, its embedding should remain relatively constant for different t. By the same logic, if a word such as `trump` does change association throughout time, its embedding should reflect this shift by moving from one position to another (*e.g.*, `estate` $\longrightarrow$ `republican`). We saw this in the previous section for static words like `president` or `mayor`; they do not change meanings, though they are accompanied by names that shift to them every few years.

To examine the quality of embedding alignment, we create a task to query equivalences across years. For example, given `obama-2012`, we want to query its equivalent word in 2002. As we know `obama` is the U.S. president in 2012; its equivalent in 2002 is `bush`, who was the U.S. president at that time. In this way, we create two testsets.

The first one is based on publicly recorded knowledge that for each year lists different names for a particular role, such as U.S. president, U.K. prime minister, NFL superbowl champion team, and so on. For each year (*e.g.*, 2012), we put its word (*e.g.*, `obama`) into the embedding set of every other year for query its equivalence in top closest words.

The second test is human-generated, for exploring more interesting concepts like emerging technologies, brands and major events (*e.g.*, disease outbreaks and financial crisis). For constructing the test word pairs, we first select emerging terms which have not been popularized before 1994, then query their well known precedents during 1990 to 1994 (*e.g.*, `app-2012` can correspond to `software-1990`). For emerging word (*e.g.*, app) we extract its most popular year (*e.g.*, 2012) with maximum frequency, and put its embedding into each year from 1990 to 1994 for querying its precedent (*e.g.*, software). Each word-year pair now forms a query and an answer; in total we have $N = 11028$ such pairs in the first testset, and $N = 445$ in the second one.

We use two metrics to evaluate the performance.

- For each test i, the correct answer word is identified at position $\text{rank}[i]$ for closest words. The *Mean Reciprocal Rank (MRR)* is defined as

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{rank}[i]}, \tag{13}$$

where $\frac{1}{\text{rank}[i]} = 0$ if the correct answer is not found in the *top*-10. Higher MRR means that correct answers appear more closely and unambiguously with the query embedding.

- Additionally, for test i consisting of a query and target word-year pair, consider the closest K words to the query embedding in the target year. If the target word is among these K words, then the *Precision@K* for test i (denoted $\text{P@K}[i]$) is 1; else, it is 0. Then the *Mean Precision@K* is defined as

$$MP@K = \frac{1}{N}\sum_{i=1}^{N}(P@K[i]). \tag{14}$$

Higher precision indicates a better ability to acquire correct answers using close embeddings.

Tables 6 and 7 show the evaluation of the alignment test. We can see that our proposed method outperforms others and shows good alignment quality, sometimes by an order of magnitude. Comparing to testset 1 which has a large amount of queries with considerable short range alignments (*e.g.*, from 2012 to 2013), the testset 2 mostly consists of fewer long range alignments (*e.g.*, 2012 to 1990). Therefore, we can see that the performance of SW2V is relatively good in testset 1 since the semantic distribution does not change much in short ranges which makes this test favorable to static embeddings. However, SW2V degrades sharply in testset 2, where the long range alignment is needed more. For TW2V, since it does an individual year-to-year (*e.g.*, 2012-to-1990) transformation by assuming that the local structure of target words does not shift, its overall alignment quality of whole embedding sets is not satisfied in testset 1 with many alignments. However, it does similarly to AW2V in testset 2 because its individual year-to-year transformation makes it more capable for long range alignment. AW2V, which enforces alignment for whole embedding sets between adjacent time slices, provides quite reliable performance. However, its alignment quality is still below ours, suggesting that their two-step approach is not as successful in enforcing global alignment.

Table 8: MRR and MP for alignment with every 3 years subsampling.

Method	r	MRR	MP@1	MP@3	MP@5	MP@10
AW2V	100%	0.1582	0.1066	0.1814	0.2241	0.2953
AW2V	10%	0.0884	0.0567	0.1020	0.1287	0.1727
AW2V	1%	0.0409	0.0255	0.0475	0.0605	0.0818
AW2V	0.1%	0.0362	0.0239	0.0416	0.0532	0.0690
DW2V	100%	0.4222	0.3306	0.4854	0.5488	0.6191
DW2V	10%	0.4394	0.3489	0.5036	0.5628	0.6292
DW2V	1%	0.4418	0.3522	0.5024	0.5636	0.6310
DW2V	0.1%	0.4427	0.3550	0.5006	0.5612	0.6299

6.3 Robustness

Finally, we explore the robustness of our embedding model against subsampling of words for select years. Table 8 shows the result of the alignment task (testset 1) for vectors computed from subsampled co-occurrence matrices for every three years from 1991 to 2015. To subsample, each element C_{ij} is replaced with a randomly drawn integer $\hat{C}_{ij}$ from a Binomial distribution for rate r and $n = C_{ij}$ trials; this simulates the number of co-occurrences measured if they had been missed with probability r. The new frequency $\hat{f}$ is then renormalized so that $\hat{f}_i/f_i = \sum_j \hat{C}_{ij}/\sum_j C_{ij}$. Listed are the alignment test results for $r = 1, 0.1, 0.01$, and 0.001 compared against [12], which otherwise performs comparably with our embedding. Unsurprisingly, for extreme attacks (leaving only 1% or 0.1% co-occurrences), the performance of [12] degrades sharply; however, because of our joint optimization approach, the performance of our embeddings seems to hold steady.

7 RELATED WORK

Temporal effects in natural language processing: There are several studies that investigate the temporal features of natural language. Some are using topic modeling on news corpus [1] or time-stamped scientific journals [6, 33, 36] to find spikes and emergences of themes and viewpoints. Simpler word count features are used in [7, 13, 22] to find hotly discussed concepts and cultural phenomena, in [21, 32, 34] to analyze teen behavior in chatrooms, and in [13] to discover incidents of influenza.

Word embedding learning: The idea of word embeddings has existed at least since the 90s, with vectors computed as rows of the co-occurrence [20], through matrix factorization [9], and most famously through deep neural networks [5, 8]. They have recently been repopularized with the success of low-dimensional embeddings like GloVE [27] and word2vec [24, 25], which have been shown to greatly improve the performance in key NLP tasks, like document clustering [16], LDA [28], and word similarity [3, 18]. There is a close connection between these recent methods and our proposed method, in that both word2vec and GloVE have been shown to be equivalent to matrix factorization of a shifted PMI matrix [17].

Temporal word embeddings and evaluations: While NLP tools have been used frequently to discover emerging word meanings and societal trends, many of them rely on changes in the co-occurrence or PMI matrix [11, 13, 22, 26, 37], changes in parts of speech, [23] or other statistical methods [4, 15, 35]. A few works use low-dimensional word embeddings, but either do no smoothing [31], or use two-step methods [12, 14, 15]. Semantic shift and emergence are also evaluated in many different ways. In [31], word shifts are identified by tracking the mean angle between a word and its neighbors. One of the several tests in [15] create synthetic data with injected semantic shifts, and quantifies the accuracy of capturing them using various time series metrics. In [23], the authors show the semantic meaningfulness of key lexical features by using them to predict the time-stamp of a particular phrase. And, [26] makes the connection that emergent meanings usually coexist with previous meanings, and use dynamic embeddings to discover and identify multisenses, evaluated against WordNet. Primarily, temporal word embeddings are evaluated against human-created databases of known semantically shifted words [12, 15, 35] which is our approach as well.

8 CONCLUSION

We studied the evolution of word semantics as a dynamic word embedding learning problem. We proposed a model to learn time-aware word embeddings and used it to dynamically mine text corpora. Our proposed method simultaneously learns the embeddings and aligns them across time, and has several benefits: higher interpretability for embeddings, better quality with less data, and more reliable alignment for across-time querying. We solved the resulting optimization problem using a scalable block coordinate descent method. We designed qualitative and quantitative methods to evaluate temporal embeddings for evolving word semantics, and showed that our dynamic embedding method performs favorably against other temporal embedding approaches.

REFERENCES

[1] James Allan, Rahul Gupta, and Vikas Khandelwal. 2001. Temporal summaries of new topics. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 10–18.

[2] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Rand-walk: A latent variable model approach to word embeddings. *arXiv preprint arXiv:1502.03520* (2015).

[3] Marco Baroni, Georgiana Dinu, and Germán Kruszewski. 2014. Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.. In *ACL (1)*. 238–247.

[4] Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. 2014. Analysing word meaning over time by exploiting temporal random indexing. In *First Italian Conference on Computational Linguistics CLiC-it*.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3 (2003), 1137–1155.

[6] David M Blei and John D Lafferty. 2006. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*. ACM, 113–120.

[7] Hyunyoung Choi and Hal Varian. 2012. Predicting the present with Google Trends. *Economic Record* 88, s1 (2012), 2–9.

[8] Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, 160–167.

[9] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391.

[10] John R Firth. 1957. {A synopsis of linguistic theory, 1930-1955}. (1957).

[11] Kristina Gulordava and Marco Baroni. 2011. A distributional similarity approach to the detection of semantic change in the Google Books Ngram corpus. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*. Association for Computational Linguistics, 67–71.

[12] William L Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. *arXiv preprint arXiv:1605.09096* (2016).

[13] Gerhard Heyer, Florian Holz, and Sven Teresniak. 2009. Change of Topics over Time-Tracking Topics by their Change of Meaning. *KDIR* 9 (2009), 223–228.

[14] Yoon Kim, Yi-I Chiu, Kentaro Hanaki, Darshan Hegde, and Slav Petrov. 2014. Temporal analysis of language through neural language models. *arXiv preprint arXiv:1405.3515* (2014).

[15] Vivek Kulkarni, Rami Al-Rfou, Bryan Perozzi, and Steven Skiena. 2015. Statistically significant detection of linguistic change. In *Proceedings of the 24th International Conference on World Wide Web*. ACM, 625–635.

[16] Matt J Kusner, Yu Sun, Nicholas I Kolkin, Kilian Q Weinberger, and others. 2015. From Word Embeddings To Document Distances.. In *ICML*, Vol. 15. 957–966.

[17] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.

[18] Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3 (2015), 211–225.

[19] Xuanyi Liao and Guang Cheng. 2016. Analysing the Semantic Change Based on Word Embedding. In *International Conference on Computer Processing of Oriental Languages*. Springer, 213–223.

[20] Kevin Lund and Curt Burgess. 1996. Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers* 28, 2 (1996), 203–208.

[21] Guy Merchant. 2001. Teenagers in cyberspace: An investigation of language use and language change in internet chatrooms. *Journal of Research in Reading* 24, 3 (2001), 293–306.

[22] Jean-Baptiste Michel, Yuan Kui Shen, Aviva Presser Aiden, Adrian Veres, Matthew K Gray, Joseph P Pickett, Dale Hoiberg, Dan Clancy, Peter Norvig, Jon Orwant, and others. 2011. Quantitative analysis of culture using millions of digitized books. *science* 331, 6014 (2011), 176–182.

[23] Rada Mihalcea and Vivi Nastase. 2012. Word epoch disambiguation: Finding how words change over time. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 259–263.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[26] Sunny Mitra, Ritwik Mitra, Martin Riedl, Chris Biemann, Animesh Mukherjee, and Pawan Goyal. 2014. That's sick dude!: Automatic identification of word sense change across different timescales. *arXiv preprint arXiv:1405.4392* (2014).

[27] Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global Vectors for Word Representation.. In *EMNLP*, Vol. 14. 1532–1543.

[28] James Petterson, Wray Buntine, Shravan M Narayanamurthy, Tibério S Caetano, and Alex J Smola. 2010. Word features for latent dirichlet allocation. In *Advances in Neural Information Processing Systems*. 1921–1929.

[29] Michael JD Powell. 1973. On search directions for minimization algorithms. *Mathematical Programming* 4, 1 (1973), 193–201.

[30] Nikhil Rao, Hsiang-Fu Yu, Pradeep K Ravikumar, and Inderjit S Dhillon. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*. 2107–2115.

[31] Eyal Sagi, Stefan Kaufmann, and Brady Clark. 2011. Tracing semantic change with latent semantic analysis. *Current methods in historical semantics* (2011), 161–183.

[32] Diane J Schiano, Coreena P Chen, Ellen Isaacs, Jeremy Ginsberg, Unnur Gretarsdottir, and Megan Huddleston. 2002. Teen use of messaging media. In *CHI'02 extended abstracts on Human factors in computing systems*. ACM, 594–595.

[33] Ruben Sipos, Adith Swaminathan, Pannaga Shivaswamy, and Thorsten Joachims. 2012. Temporal corpus summarization using submodular word coverage. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 754–763.

[34] Sali A Tagliamonte and Derek Denis. 2008. Linguistic ruin? LOL! Instant messaging and teen language. *American speech* 83, 1 (2008), 3–34.

[35] Xuri Tang, Weiguang Qu, and Xiaohe Chen. 2016. Semantic change computation: A successive approach. *World Wide Web* 19, 3 (2016), 375–415.

[36] Xuerui Wang and Andrew McCallum. 2006. Topics over time: a non-Markov continuous-time model of topical trends. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 424–433.

[37] Derry Tanti Wijaya and Reyyan Yeniterzi. 2011. Understanding semantic change of words over centuries. In *Proceedings of the 2011 international workshop on DETecting and Exploiting Cultural diversiTy on the social web*. ACM, 35–40.

[38] Stephen J Wright. 2015. Coordinate descent algorithms. *Mathematical Programming* 151, 1 (2015), 3–34.

[39] Hsiang-Fu Yu, Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. 2012. Scalable coordinate descent approaches to parallel matrix factorization for recommender systems. In *12th IEEE International Conference on Data Mining (ICDM)*. IEEE, 765–774.

[40] Yating Zhang, Adam Jatowt, Sourav S Bhowmick, and Katsumi Tanaka. 2016. The Past is Not a Foreign Country: Detecting Semantically Similar Terms across Time. *IEEE Transactions on Knowledge and Data Engineering* 28, 10 (2016), 2793–2807.