Improving Semantic Change Analysis by Combining Word Embeddings and Word Frequencies

Adrian Englhardt · Jens Willkomm · Martin Schäler · Klemens Böhm

Received: 04 May 2018 / Revised: 22 March 2019 / Accepted: 07 May 2019

Abstract Language is constantly evolving. As part of diachronic linguistics, semantic change analysis examines how the meanings of words evolve over time. Such semantic awareness is important to retrieve content from digital libraries. Recent research on semantic change analysis relying on word embeddings has yielded significant improvements over previous work. However, a recent, but somewhat neglected observation so far is that the rate of semantic shift negatively correlates with word-usage frequency. In this article, we therefore propose SCAF, *Semantic Change Analysis with Frequency*. It abstracts from the concrete embeddings and includes word frequencies as an orthogonal feature. SCAF allows using different combinations of embedding type, optimization algorithm and alignment method. Additionally, we leverage existing approaches for time series analysis, by using change detection methods to identify semantic shifts. In an evaluation with a realistic setup, SCAF achieves better detection rates than prior approaches, 95% instead of 51%. On the Google Books Ngram data set, our approach detects both known as well as yet unknown shifts for popular words.

Keywords computational linguistics · semantic change analysis · change detection · word embeddings

1 Introduction

Language is in constant flux. Humans seek efficiency when expressing themselves and communicate according to the least-effort principle proposed by Zipf [1]. So new words are invented or adapted, while others vanish. To illustrate, "Katrina" as a hurricane name suddenly started appearing in 2005. In order to support retrieval of content from digital libraries, sophisticated search mechanisms have become available in the last decades [2]. Managing the resources of a digital library requires knowledge of the meaning of words. A system for search should be aware of semantic changes, to better support a user. In particular, when searching in historical corpora, the meanings of words that have changed must be taken into account.

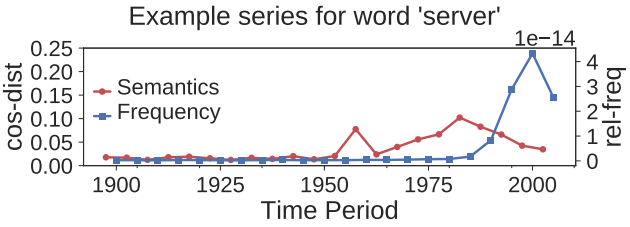

Fig. 1 Example time series for the word "server" with semantic and frequency information. The *red* series represent the amount of semantic shift measured in the embeddings while the *blue* visualizes the usage of the word.

Diachronic linguistics studies the development of a language over time. Nowadays, digital content is available that contains temporal information. Such data offers new research possibilities for that branch of linguistics. A commonly used data set for diachronic analysis is the Google Books Ngram corpus [3]. Together with advances in distributional language models, new methods have recently been proposed to analyze semantic shifts computationally [4–7]. While some methods only extract the words whose meanings have shifted, diachronic linguistics seeks to identify the words together with the points in time when the shifts occurs [4, 8]. Recent work also has analyzed social media data such as

Adrian Englhardt
Karlsruhe Institute of Technology, Germany
E-mail: adrian.englhardt@kit.edu

the Twitter stream or news articles, in order to identify semantic shifts [5, 9–11].

Problem Statement Previous work has focused on extracting words that have changed semantically in language models, e.g., in word embedding models. Based on the models, the words that have changed most have been computed. Recent results in [4] indicate that the rate of semantic shifts negatively correlates with frequency. I.e., frequent words change more slowly. When working with word embedding models, without any filtering of the vocabulary, infrequent words dominate the top lists of words that change most. This is due to their frequently changing surrounding context. Additionally, words that gain or change to new meanings tend to be used more often. In this article, we study how explicitly including word frequency information can improve semantic shift analysis. Our evaluation focuses on words with semantic shifts where the frequency is stable or increases. We discuss semantic shifts where frequency decreases only briefly in Section 4.2.2.

Example 1 The word "server" did describe the role of an assistant at the start of the 20th century. In the 1950s it appeared in queue theory [12], until it shifted towards its current meaning mixture in the late 20th century. This is in line with the red line in Figure 1. Later, the popularity of the word has increased considerably, as is evident from the increasing relative frequency, the blue line in Figure 1. Most existing methods for semantic shift analysis would only return the word "server", because of the fluctuations of the red curve, but without points in time. Next, a method featuring change detection but leaving aside frequencies would for instance return the point in time of the second local maximum of the red curve. However, observe the steep increase of the blue curve shortly before the year 2000. It occurs because, with its new meaning, the word is used more often. Inspecting the actual data reveals that "server" is indeed used in this way. At the same time, the new meaning needs some time to gain momentum, so the relevant actual point in time is after the maximum of the red curve. Our method indeed positions the point in time there.

Challenges The problem is challenging in several ways: A core question is how frequency information can actually be used for semantic shift detection. Plain filtering of the vocabulary a priori is insufficient to handle the different degrees of semantic shifts. Instead, semantic and frequency information must be considered simultaneously. Deploying approaches from time series analysis seems to be more promising, but this is not trivial. This is because it requires a time series representation that covers both kinds of information. Working with word embeddings for diachronic linguistics is challenging as well: Due to the nonlinear optimization, the use of approximation approaches and random initialization of the learning process, different runs of the training algorithm on a corpus can result in different word vectors. When applying transformations [4, 5, 13], or when training incrementally [4, 6, 14], the models need to be aligned, to enable comparisons. However, the influence of the different alignment approaches has not yet been studied, and this increases the parameter space for word embeddings even further. Another challenge is the lack of a ground truth of words that have shifted semantically in a language. To allow for a quantitative evaluation, synthetic corpora are needed that reflect the gradual nature of semantic shifts. This is different from abrupt changes, as for instance modeled in [5].

Contributions In this work, we propose an abstraction from the concrete embeddings that takes word frequency information into account: SCAF, *Semantic Change Analysis with Frequency*. We combine information on the movement of a word in the embedded space over time and its occurrence count to a two-dimensional time series model. This allows to apply any multidimensional change detection algorithm, to detect and quantify semantic shifts. Our extension of the time series model with frequency is orthogonal to specifics of the embeddings. Put differently, our approach is universal, meaning that it works with any combination of word-embedding type, approximation algorithm and alignment. This allows to benchmark all combinations. We show that our method is superior in several ways: Our method benefits from the inclusion of frequency and outperforms existing approaches. In a realistic setup, the detection rate improves by more than 40%, and change detection run time is in the order of seconds instead of hours. We compensate for popular words shifting more slowly and identify yet unknown shifts in the Google Books Ngram data set.

SCAF can be applied to any temporal data, e.g., to the Google Books Ngram data set or to web crawls. While granularity obviously is different, the patterns, e.g., 5-year periods over two centuries versus monthly data over 5 years, are very similar. In this article, the main focus is on the Google Books Ngram data for our experiments. Previous approaches have focused on this data as well. So this allows for comparisons of results. This data set is one of the most comprehensive historical corpora and has over 1 TB uncompressed. Additionally, we apply our approach on Twitter data in Section 5.4.

We publish all material and refer to our website[1] for a description of the data. The material includes all trained word embedding models which have had a total training time of several weeks on modern machines. The material should reduce the technical effort of other researchers analyzing semantic shifts as well.

2 Change detection

We now briefly review change detection and its properties that are relevant for our approach. We present the requirements on the change-detection method and the change-detection method of our choice. The change-detection method is an integral part of our approach. So we present it right here and not as part of the section on related work, in order to highlight its importance.

Different approaches exist to identify changes in temporal data. One option is to encode the detection directly as part of a "larger", domain-specific analysis. In our case, this would be the detection of words that have shifted. For instance, work in [4, 6] uses the summed vector movement to sort the vocabulary where the real shift time remains unknown. The alternative is to leverage existing approaches and implementations for time series analysis, i.e., use so-called change detection algorithms. This, i.e., factoring out common functionality in order to be able to focus on the application-specific one, is common practice in computer science. We now give a definition of the general change-detection problem for an arbitrary time series. We follow this definition when including change detection into our analysis pipeline later on.

Definition 1 (Change detection) Given a time series with values $x_t \in \mathbb{R}^n$ with points in time $t \in \mathcal{T}$ as input, *change-detection* is a function which maps $\mathbb{R}^{n \times |\mathcal{T}|}$ to a set of changes $c \in \mathcal{T} \times \mathbb{R}$.

Executing *change-detection*(ts) on a time series ts yields a set of changes as tuples $(\hat{t}, \hat{s})$. Here, $\hat{t}$ is the point in time of the change, and $\hat{s}$ is an optional score.

One can categorize change detection methods along different dimensions: One is the dimensionality of the time series, i.e., one- or multi-dimensional. Another one is the type of change the method detects, e.g., shift in mean or variance. One more distinction is that methods may or may not rely on assumptions regarding the value distribution and properties of a time series. Next, a time series is homogeneous when the n dimensions of each value x_t are at exactly the same point in time. Distinguishing whether a method works on inhomogeneous time series is another dimension. See [15] for an overview.

[1] `https://dbis.ipd.kit.edu/2601.php`

Our case has the following requirements on change detection:

(a) Fast run time, to facilitate execution on a full vocabulary
(b) Multi-dimensional, to allow the combination of semantic and frequency information
(c) Detect a shift in mean when a context of a word or a frequency shifts
(d) Work on short time series: The time series from, say, the Google Books Ngram data set for 1800–1999 only have 40 values with 5-year periods

A study of different change detection methods in terms of performance or quality in this current context is beyond our scope and is not the focus of this work. Instead, we study how good approaches based on a conventional method actually are (with very positive results, as we will see). We select a well-known change detection approach based on cumulative sums, known as CUSUM [15, 16]. CUSUM fulfills the above requirements: Regarding (a), the calculation only requires two passes over the time series. Its parallelization is trivial, by calculating changes for several series at once. CUSUM can work with multi-dimensional time series (b). With an appropriate time series model, the cumulative sum captures changes in mean (c). It also does so when the series are short (d).

A key characteristic of CUSUM is that the summed time series must only contain change information, e.g., a time series $x_t \in \mathbb{R}^n$ of differences. Formally, the equations for the sum S_t are the following ones, when all dimensions $j \in [1, \ldots, n]$ are treated equally:

$$S_0 = 0, \qquad S_t = S_{t-1} + \sum_{j=1}^{n} x_{t,j} \tag{1}$$

Then one can identify the time of the most significant shift and score $(\hat{t}, \hat{s})$ in the series by computing:

$$(\hat{t}, \hat{s}) = \Big(\arg\max_t(S_t), \max(S_t)\Big) \tag{2}$$

CUSUM can be adapted to work with inhomogeneous time series. But we want to preserve the flexibility to apply other change detection methods and hence build a homogeneous model.

3 Related Work

We now review word embedding models to study the evolution of a language, then semantic change analysis. Finally, we present change detection methods used for semantic shift detection.

Word embeddings in diachronic linguistics Distributional models are used for tasks in NLP, e.g., examining probability distributions [17, 18], generating video descriptions [19] or grammar/spell checking [20, 21]. Word embedding models build low-dimensional, real-valued representation of a vocabulary where similar words are close to each other. Several approaches to learn embeddings exist, such as Positive Pointwise Mutual Information (PPMI), Singular Value Decomposition (SVD), Latent Semantic Analysis (LSA), Skip-gram (SG) and Continuous Bag of Words (CB), published in the *word2vec* toolkit [22], or GloVe [23]. CB and SG optimize the embeddings by looking at pairs of words that occur together and some that do not. Since using the full vocabulary for negative pairs is infeasible, so-called approximation algorithms are in use for the optimization. *Hierarchical softmax* (HS) and *negative sampling* (NS) [24] are well-known approaches that solve the optimization problem.

Another common problem of all embedding models just presented is that the embeddings for different time periods are not directly comparable. Learning two embedding models from exactly the same data can result in completely different word vectors. The vectors can be shifted and rotated. Related work therefore suggests alignment transformations based on linear problems [5, 25], on orthogonal Procrustes [4, 9] or training incrementally [4, 6]. For the latter, the word vectors of time period t are initialized with the ones of the previous time period $t-1$. Only the first time period is initialized randomly. While new embedding approaches exist that try to overcome the alignment issue directly such as Temporal Random Indexing (TRI) [8] or others [11, 26], we focus on the well-established methods of the *word2vec* toolkit, i.e., combinations of SG/CB and HS/NS. The respective abbreviations are SGHS, SGNS, CBHS and CBNS in the following. Nevertheless, our approach is universal in that it works with any underlying word embedding model, as we will describe in Section 4.1.

Previous work suggests that measuring model quality solely with commonly used word sense or analogy test sets is error-prone and does not always identify the best combination for down-stream NLP tasks [27, 28]. This problem often also is referred to as intrinsic versus extrinsic evaluation. "Intrinsic" corresponds to the direct syntactic or semantic relationships between words in the models and relies on intellectual assessment [9]. "Extrinsic" evaluation can, in our case, take place by testing the performance on the semantic shift detection task in a quantitative manner. By doing so, one can select the best possible combination without being misguided by individual model quality.

Hellrich et al. have tried to answer the combination problem of different word embedding types and approximation algorithms, but with a focus on short time periods (1900–1904) and only with intrinsic evaluations [29]. Another issue is the parameterization of the embedding models. Various parameters must be configured before training an embedding model such as SGNS, e.g., the vocabulary size, embedding dimensionality or window size. Previous work exists that studies the influence of the parameters on the models used in this current article [30, 31]. Next, we ourselves have already studied the training of embedding models from ngrammed contemporary corpora in [32]. In this article, we use findings from this previous work in order to shrink the parameter space. However, all those studies focus on embedding models trained on contemporary language and not on historical corpora. This article in turn features an exhaustive intrinsic and extrinsic study of training embeddings from historical corpora for semantic shift analyses, see Section 5.1 and Section 5.2.

Semantic shift analysis A prominent research topic in NLP is the evolution of language. Early work has focused on smaller case studies of a few words and has often required linguistic knowledge [33–39]. Together with the advances in distributional language models, efficient implementations and the publication of the Google Books Ngram corpus [3, 40], others have designed methods to compute semantic shifts. While an early approach is based on co-occurrence matrices [41], work has then relied on word embeddings to extract the evolution of word semantics [5, 6].

Kim et al. have extracted words with the largest movement in incrementally trained SGNS models [6]. Kulkarni et al. have used change detection to identify change points on linearly aligned SGNS embeddings [5]. The law of conformity has been presented by Hamilton et al. in [4] with an analysis of the correlation between the movement of words in vector space compared to frequency or polysemy. Most recent work includes external resources such as WordNet [42] or has performed refined analysis for other languages [25, 43–45]. We are the first to use the results of [4] to improve the detection quality for popular words that shift more slowly.

Named entity evolution is another research direction that studies the evolution of language. Here, the objective is different from what we have previously introduced as semantic shift: For named entity evolution, the context remains similar, and the names of an entity change. On the other hand, in semantic shift analysis as we see it, the term itself is fixed, and it is the context that changes. Named entity evolution relies on the comparison of context words. Doing so for the cross

Table 1 Existing time series models for shift analysis.

Embed.	Alignment	Measure	Analysis focus
LSA	-	cos-sim	similar word pairs [7]
SGNS	incremental	cos-sim	most changed words [6]
SGNS	local-linear	cos-dist	change detection [5]
SGNS	incremental	cos-dist	most changed words [4]
SVD	Procrustes	cos-dist	most changed words [4]
TRI	-	cos-sim	change detection [44]

product of all words is computationally expensive [46]. Tahmasebi et al. solve this problem by filtering with a burst detection [46]. Only time windows and words are analyzed where word frequency has increased by much. This filtering technique however misses shifts where frequency only changes slowly or not at all. Our method in turn is able to detect such changes, by always analyzing the full dictionary and the full time span.

Change detection for semantic shift analysis Kulkarni et al. used a method based on a mean shift model that estimates the change probability by pivoting the series at each element and calculating the mean for both parts [5]. Since the method relies on random permutations for the estimation, it is computationally expensive. Nevertheless, we will include it for comparison, since it is the state-of-the-art change detection method for semantic shift analysis. Work in [44] also has applied this change detection approach. In this article, we propose an approach with a better detection rate as well as a shorter run time.

We see [5] as our main competitor, and our approach consists of similar building blocks. However, it features several new contributions. In this current article, we propose a new two-dimensional time-series model that directly includes the frequency information. We use a different change-detection method that can process multidimensional time series. In Section 5.1 we assess the quality of the word embeddings trained on historical corpora which to this extent is missing in related work. Finally, our approach detects more plausible change points than previous work in a realistic setup, as we will see in Section 5.2.

4 Our Abstract Time Series Model

As introduced in Section 3, previous work has used different approaches to perform semantic shift analysis with word embedding models. Table 1 is a summary of some existing time series models. It should become evident that the parameter space is enormous, and that research has followed a variety of directions. Our contribution starts by unifying these approaches and by proposing an abstracted model. We then continue with an orthogonal extension by including frequency information. The notation used in this article is featured in Table 2.

Table 2 Notation used in this article

	Explanation
$\mathcal{T}$	time periods
$\mathcal{V}$	vocabulary
w_i	word i in $\mathcal{V}$
$\mathcal{C}_t$	corpus for time period t
ϕ_t	embedding model trained for time period t
$\phi_t(w_i)$	word vector for w_i at time t
af_t	absolute frequency of a word at time t
f_t	relative frequency of a word at time t
s_t	cos-dist between the vectors of a word at $t-1$ and t

4.1 Abstraction

The following abstraction is the basis for our approach.

Structural Aspects: Given a vocabulary $\mathcal{V}$ and temporally consecutive corpora $\mathcal{C}_t$, one learns word embeddings $\phi_t \colon \mathcal{V}, \mathcal{C}_t \mapsto \mathbb{R}^d$ that map each word to a d-dimensional vector. Depending on the training procedure, the embeddings are not always directly comparable when learning different models for each time period. Nevertheless, we assume that the models ϕ_t are aligned, e.g., with approaches presented in Section 3. But we do not have any restriction on how the actual alignment is performed. We compare several alignment approaches in Section 5.2.

Measuring Semantics: Given a sequence of word embeddings, it is possible to compute values

$$\begin{aligned} s_t &= \text{cos-dist}(\phi_{t-1}(w_i), \phi_t(w_i)) \\ &= 1 - \frac{\phi_{t-1}(w_i) \cdot \phi_t(w_i)}{||\phi_{t-1}(w_i)|| \cdot ||\phi_t(w_i)||} \end{aligned}$$

for each word $w_i \in \mathcal{V}$. They represent the movement of a word in the space. The values s_t form a time series, which allows for semantic shift analysis.

The relevant word embedding parameters in this article are:

- word embedding model (e.g., CB, SG, SVD, PPMI, LSA, etc.)
- approximation algorithm (e.g., HS, NS)
- alignment approach (e.g., incremental training, linear or Procrustes based transformation, etc.)

For brevity, we speak of "combination" when we talk about combining different word embedding types, approximation algorithms, and alignment methods. Algorithm 1 represents the full analysis pipeline from the

Algorithm 1: Analysis Pipeline

```
1  Input: Data set 𝒟, change-detection, embedding
       type emb_type, approximation algorithm
       emb_approx, alignment approach emb_align
   Result: changes = [(w, t, s), ...] for each word
2  C_1, ..., C_k = split-and-clean(𝒟)
3  𝒱 = vocabulary(𝒟)
4  for C_i ∈ C_1, ..., C_k do
5  |  φ_i = train-embedding(emb_type, emb_approx, C_i)
6  end
7  φ̃_1, ..., φ̃_k = align-embeddings(emb_align, [φ_1, ..., φ_k])
8  for w_i ∈ 𝒱 do
9  |  ts = build-time-series(w_i, [φ̃_1, ..., φ̃_k], [C_1, ..., C_k])
10 |  (t̂, ŝ) = change-detection(ts)
11 |  changes.append(w_i, t̂, ŝ)                    SCAF
12 end
```

raw data set, training and aligning the embeddings, to the changes. First, function *split-and-clean* preprocesses the data set; we will discuss it in Section 5. Function *train-embedding* then trains an embedding model. We use *align-embeddings* when aligning the embeddings. Function *build-time-series* builds our new time series model, which we introduce in the following section. Based on it, we then detect changes in Line 10 and retrieve the most significant change for each word, together with a score.

SCAF can work with arbitrary word embedding models as long as they are aligned. If the aligned embedding models are given externally, we can skip the first part of the analysis pipeline. Put differently, one can hot-start our method in Line 8 with the following input: aligned embedding models $\tilde{\phi}_i$, vocabulary $\mathcal{V}$, and corpora $\mathcal{C}_i$. This also includes the case when working with a joint embedding and alignment model like the ones mentioned in Section 3. We describe this straightforward abstraction explicitly since previous work has focused on individual combinations and does not feature any comprehensive comparison.

4.2 Universal Extension

We use the abstraction just presented to extend the time series model with word frequency information. In general, the extension yields a two-dimensional time series where the first dimension contains semantic information and the second one frequency data. As we will explain later, the extension is not just an obvious combination. This is because homogenizing the dimensions is not trivial.

From the abstracted model, we retrieve the values s_t which quantify the movement of a word in the vector space over time. For the second dimension, we first collect the absolute frequencies $af_t(w_i)$, by counting the occurrences for each $w_i \in \mathcal{V}$ per corpus $\mathcal{C}_t$. In order to extract information from these raw frequency values, they need to be comparable over time. Word occurrence fluctuates in the Google Books Ngram data set, since the numbers of digitized books per year differ. Additionally, the data set contains an exponentially increasing number of words, for the latest 50 years in particular [3]. So we follow the creators of the data set and use the relative frequency [40]. Each occurrence count is divided by the total count of words in the corresponding time period $|\mathcal{C}_t|$, and we retrieve the value $f_t = af_t(w_i)/|\mathcal{C}_t|$ per word w_i.

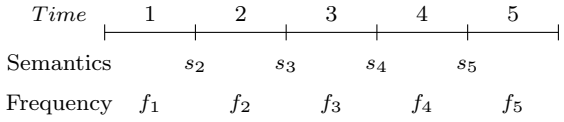

Fig. 2 Time series model with dimension combination.

We now go over remaining open issues. Afterwards, we present our model that combines semantic and frequency information.

4.2.1 Dimension Combination Problem

The modeled values s_t and f_t correspond to different positions in time, as Figure 2 shows. The semantic value s_t of a word represents how the word vector moves from time period $t-1$ to period t. The frequency values f_t in turn directly correspond to period t. To preserve the flexibility to apply other standard change detection methods, the time series must be homogeneous. Therefore, the dimensions must be adjusted, i.e., for each time period there should be one value for semantic and one for frequency information. This is not yet the case in our setting. Additionally, we have to consider the different value ranges for s_t and $f_{t-1,t}$ when combining changes in both dimensions. Formally defined, the homogenization required is the following one: Given the semantic values $SV = (s_2, \ldots, s_t)$ and frequency values $FV = (f_1, \ldots, f_t)$, we seek a homogenization $hom(SV, FV) = (x_2, \ldots, x_t)$ with $x_i \in \mathbb{R}$ for all $i \in [2, \ldots t]$. The models presented in the following section will follow this scheme.

4.2.2 Models

We have decided to use the CUSUM change detection to extract the most significant change point in each series. We also experimented with a CUSUM version where a change is only detected when a certain threshold parameter is exceeded. Due to value fluctuations in the series, the choice of this value proved to be difficult, and it does not generalize when switching to other corpora.

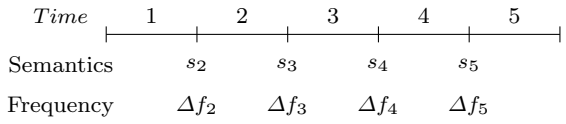

Fig. 3 Time series model A.

This is why we choose the non-parametric CUSUM version presented in Section 2. Applying CUSUM to our time series model yields a ranked list of the words from the vocabulary where the scores indicate potential semantic shifts.

The choice of CUSUM as the change detection algorithm imposes two conditions on the dimension and value adjustment: Our time series model must only contain change information, and both dimensions must have similar value ranges. For a homogeneous time series, a two-dimensional value x_t per time period t is required that combines the semantic information values s_t and frequency f_t.

To ease presentation, we now present a first model to carve out the requirements a model indeed has to meet. In what follows, the word "model" always stands for the time series model envisioned for semantic shift analysis including word frequency.

Model A The first model uses the raw cosine distances s_t in the first dimension. With the cosine distance, we sum over the movement of a word in the vector space, and a high movement indicates a semantic change. For the second dimension, taking the actual relative frequency values is not an option. This is because they do not reflect any change information, and summing them up is meaningless. Instead we use the differences, by computing $\Delta f_t = f_t - f_{t-1}$. Figure 3 visualizes the model. The values for the final time series are $x_t = s_t + \Delta f_t$.

This model comes with several problems: In the first dimension, the values s_t generally are greater than zero and range at about 0.01–0.15. The word vectors are not perfectly stable even without any semantic shift. Experimentally this can be explained by the sampling bias and noise in the data sets. When computing a cumulative sum over the values s_t, it is not possible to extract a change point. This is because the sum is steadily increasing. Normalization is difficult as well. This is because the value distribution depends on the word, e.g., the values for "the" hover around 0.015 and for the word "car" around 0.035 with incrementally trained SGNS embeddings. Applying a normalization such as the z-normalization on each series individually is not an option. This is because the change detection is sensitive to deviating values. A global z-normalization of all series at once does not fix the issue either, as seen from the example words "the" and "car".

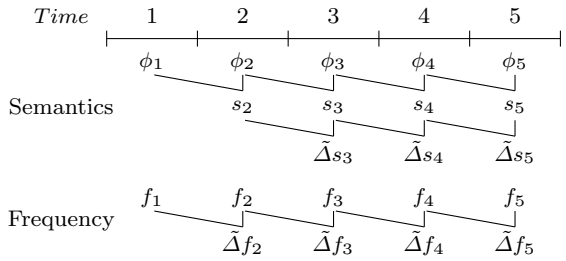

Fig. 4 Time series model B.

Regarding the second dimension, the deltas for the frequency reflect increasing or decreasing popularity of a word. This may indicate a semantic shift. But the value ranges of s_t and Δf_t are too different and require additional weighting when computing the sum per time period. While s_t commonly ranges between 0.05–0.2, f_t is a lot smaller, e.g., 0.0001–0.05. By additionally computing deltas Δf_t for the latter, the difference between the ranges further increases, and the values are at least a magnitude smaller.

Model B We now present a second model which copes with these problems. Instead of using the raw cosine distance value s_t, we also calculate differences $\Delta s_t = s_t - s_{t-1}$. To illustrate, the value Δs_3 indicates whether the vector of a word moved more or less from time point 2 to 3, as compared to time points 1 and 2. By doing so, we cope with the instability of the word vectors. We get rid of continuous movement caused by noise in the data set. At the same time, we are sensitive to a word suddenly moving more. For the second dimension we keep using the deltas for the frequency Δf_t. While one might consider a formula similar to the one for the cosine distances to compute a second order delta, this is not meaningful. This is because a constant positive value for Δf_t already indicates that a word is gaining more and more popularity, i.e., these values already are meaningful input for a change detection algorithm.

In order to balance the value range discrepancy for the two dimensions, we modify the delta calculation slightly and compute the percentage difference. Based on the cosine distance values s_t and frequency values f_t, the percentage deltas then are as follows:

$$\tilde{\Delta}s_t = \frac{s_t}{s_{t-1}} - 1, \qquad \tilde{\Delta}f_t = \frac{f_t}{f_{t-1}} - 1 \tag{3}$$

If the goal is to search for semantic shifts where frequency decreases, we can modify the formula for the frequency change to $\tilde{\Delta}f_t = \frac{f_{t-1}}{f_t} - 1$, i.e., the inverted fraction. The model then is sensitive to frequency decreases with $f_{t-1} > f_t$. In this article however, the focus

of the evaluation is on semantic shifts where frequency is stable or increases. So we stick to the formula given in Equation 3.

The last open question is how to handle the dimensional displacement. Starting with the second time period, we assign $\tilde{\Delta}s_t$ and $\tilde{\Delta}f_t$ to period t. The resulting time series is $x_t = \tilde{\Delta}s_t + \tilde{\Delta}f_t$. This allows to directly sum the new deltas and to perform a change detection based on cumulative sums. Figure 4 graphs the model.

5 Evaluation

The evaluation chapter is structured in a bottom-up manner:

- Intrinsic evaluation of training word embedding models from the Books data set
- Extrinsic evaluation on the task of detecting semantic shifts on a synthetic corpus
- Exploration of the Books and Twitter data set

In Section 5.1, we perform experiments on the performance of word embeddings on the Google Books Ngram data set. Previous work has only tested this superficially [29]. To this end, we first target at the best combination for the Books corpus. In Section 5.2, we construct a synthetic corpus and show that our method extracts the manually induced shifts and outperforms existing approaches. With this knowledge we turn to the Books corpus in Section 5.3 and Twitter corpus in Section 5.4 and apply our model.

Google Books Ngram data set As part of Google's Books initiative to provide a searchable resource, the Google Books Ngram data set has been published [3]. In this article, we focus on the revised second version and its English 5gram part. To decrease the number of models required to train, we group the data to 5-year periods, as a trade-off between yearly analysis [6] and decadewise [4]. With this aggregation, the *match count* values are summed. We use the time period from 1750–1799 as initialization when training incrementally and focus our analysis on 1800–2008. We clean the raw data by ignoring the part of speech annotation of Ngrams and by removing any punctuation or numbers, followed by lower casing. The splitting to 5-year periods and the data cleaning is the *split-and-clean* function in Algorithm 1. The resulting corpus is still of enormous size, with over 100 GB compressed.

Twitter data set As an additional data set, we use the 1% sample of the Twitter stream available in the repositories of the Internet Archive[2]. Twitter data is a second data set with properties different from the ones of the Google Books data . This is because the data is from a social network, and tweets have been limited to the 140 characters at that time. In this work, we use the data from Januar 2016 until June 2017. We aggregate the data by month and filter all non English tweets with the help of the available language tag. We then remove any punctuation, URLs and other special characters. We only remove the "#" symbol in hashtags. By doing so, we preserve the general semantics because hashtags are often used in the middle of sentences. Finally, we lower case the text. These steps are the *split-and-clean* function in Algorithm 1. The resulting corpus is around 15 GB compressed.

For both real world corpora we follow related work to clean and filter the raw text, in order to achieve comparability of results. To illustrate, the Books corpus is additionally filtered with the word list provided in [4] during the exploration in Section 5.3.2. We note that words that appear as proper nouns later are still part of the analysis. To our knowledge, no word lists are available to filter such words. Managing such a list would be cumbersome for the very recent Twitter data in particular.

5.1 Intrinsic Embedding Parameter Evaluation

To analyze semantic shifts with word embeddings, models are required that correctly represent the language. Therefore, we first investigate how to train good distributional language models on the Google Books Ngram data set. This kind of evaluation is often referred to as part of *synchronic* linguistics that inspects the language at a given point in time. It is opposed to *diachronic* linguistics which investigates how and why a language changes over time. We seek answers to the following questions:

1. Match count: How to use the frequency (i.e., *match count* in the Books data format) of the Ngrams during model training?
Rationale behind the question: Previous work on semantic shift analysis on the Google Books Ngram data set does not state how the *match count* of each Ngram is used.

2. Corpus size: Is the commonly used number of 1 billion words [24] enough when working with the Ngram format?
Rationale behind the question: Training models from full-text corpora is well studied but Google Books Ngrams have a restricted context of five words.

3. Historical data: How do models from historical data perform? Are tests for contemporary language applicable to older versions of the language?

[2] `https://archive.org/details/twitterstream`

Rationale behind the question: Training word embedding models on contemporary language has been studied [22–24]. For semantic shift analysis, we require good models for historical data though.

We now examine the parameters for the word embeddings. Afterwards we go over the results and discuss them.

5.1.1 Parameters

Word embeddings We do not want to rely on results gained on training distributional models on contemporary language since historic corpora are different in size and quality. This is why we evaluate the performance of different embedding combinations. We choose the models *Continuous Bag of Words* (CB) and *Skip-Gram* (SG) together with the approximation algorithms *hierarchical softmax* (HS) and *negative sampling* (NS). We refer to the original papers [22, 24] for an explanation.

We use the *gensim* [47] implementation which provides common default values. The target vector dimension is set to 200, and we train five iterations on one billion words unless stated otherwise. This is because previous work of ours has yielded good models for these values [31, 32].

Match count handling Another important decision is how to handle the *match count* of the Ngrams. Previous work often does not state how the authors have proceeded. But when sampling to a maximum of one billion words in particular, the decision is crucial when looking at our results. We therefore compare different approaches and call the parameter *corpus building mode*. The *standard* approach is to duplicate each Ngram according to the *match count*. The second option is called *scale* and scales the *match count* by dividing by its quadratic logarithm. We call the extreme approach *ignore*. It completely drops the count, and the embedding model only sees each Ngram once per iteration.

So the target parameter space that we evaluate is the following one: {CB, SG} × {HS, NS} × {standard, scale, ignore} × {1 billion, 5 billion}. For the third question, we additionally evaluate the performance on historical data. For this part of the evaluation we train over 500 embedding models. Each model requires 2–16 hours training time, using 4 cores on a Intel®Xeon™E5-2630 v3 at 2.40GHz.

5.1.2 Benchmarks

We conduct two different types of tests of our word embedding models. The first one is generally referred to as *analogy* test and evaluates how well a model can capture analogies such as "king is to man what queen is to woman". We use the test set published together with the *word2vec* toolkit and one from Microsoft Research collected in [4, 5]. The second type is more relevant for semantic shift analysis since it evaluates the similarities between words based on lists humans have created. We refer to this type as *word sense* test and use six test sets collected in [4, 5]. For scoring, we follow related work by averaging the accuracy for *analogy* tests. For the word sense tests, we calculate the correlations between the similarities of the model and the ones given by human judges.

As a reference, we choose a trained model published by the author of the *word2vec* toolkit[3]. The model serves as a reference and not as a baseline. This is because the SGNS combination was trained on about 100 billion words, and the model contains three million words. This is considerably more than what we work with. In plots that will follow, the scores for the reference model will appear as dashed lines.

5.1.3 Results

We now present our answers to the questions.

Corpus building mode. We first evaluate the different approaches for handling the *match count*. The results for different combinations of word embedding types and approximation algorithms are in Figure 5. The models are trained on a contemporary sample of the Books data set from 2000–2004. We see that NS outperforms HS, and the gap is particularly noticeable on the analogy tasks. The standard approach of using the *match count* yields models of very inferior quality. Together with the corpus size of one billion words, the text learned during training is less diverse. This is because Ngrams with high *match count* fill up the desired number of words quickly. Quality significantly increases by scaling or even completely ignoring the frequency. The achieved scores are substantially higher. On the other hand, model training time also increases. This is because during the training the corpus must be scanned in a deeper manner to reach the configured maximum number of words.

Corpus size. Another option to improve the quality of the corpus learned by the embedding model is increasing the maximum number of words used. As mentioned, one billion words are commonly used. But the previous paragraph indicates that this number may not be adequate when working with Ngrams. In Table 3 the results for CBNS and SGNS for one billion and five billion words are compared for all three ways of handling the *match count*. While the standard approach certainly

[3] `https://code.google.com/archive/p/word2vec/`

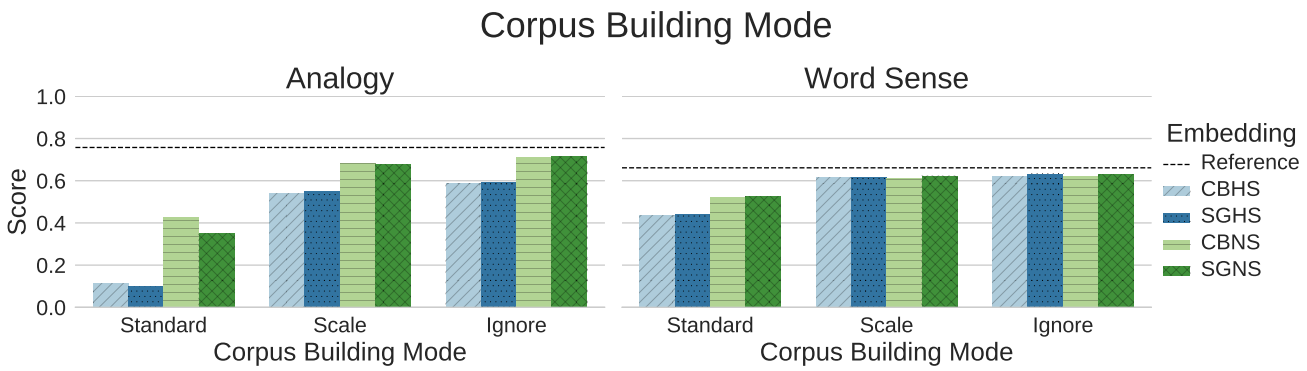

Fig. 5 Scores for different approaches of handling the *match count* for different embedding combinations with the data from the Books corpus from 2000–2004.

Table 3 Averaged word sense scores for 1 billion and 5 billion corpus size.

		Corpus building mode		
	Corpus Size	Standard	Scale	Ignore
CBNS	1 billion	0.522	0.612	0.621
	5 billion	0.580	0.619	0.620
SGNS	1 billion	0.527	0.623	0.631
	5 billion	0.592	0.637	0.640

profits from the additional data, the improvements with scaling or ignoring the *match count* are very small. Furthermore, training time increases as well. This means that the corpus which is five times larger requires at least twice the training time.

Historical data. As the last part of this section, we analyze the quality of training models from the historical Books data, i.e., comparing 1750–2008 for CBNS and SGNS. We train a model for each 5-year period and benchmark with the two test types presented. The results are in Figure 6 for both combinations with NS. The scores converge towards the performance of models trained on the most recent data. This also represents the state of the language the tasks were designed for. The scores on the analogy tests roughly reach the performance of models trained on contemporary language. Word sense benchmarks are more sensible to historical data, in scores from data before 1900 in particular. A closer look reveals that the number of missing words is the reason. The models from the 18th century are unable to answer about 15–35% of the word sense questions due to words not being in the trained vocabulary and not even in the corpus itself. This number drops to about 5–10% at the start of the 20th century and is at about 3% with the most recent data.

To further investigate the influence of the missing words on embedding performance, we filter the language tests to only include words that are present in all 5-year time periods. The resulting scores are slightly higher with the filtering than without. Yet the behavior of the scores over time is similar to the one presented in Figure 6. The performance of the models in the 18th century is subpar, and the scores converge to the ones of the reference model for the most recent corpora. The difference between the scores with unfiltered and filtered test sets is higher for the word-sense tests. This is because the share of missing words in the unfiltered setting is higher for the word-sense tests than for the analogy tests as well.

5.1.4 Discussion

In this section, we have analyzed how to train a large number of models for semantic shift analysis with respect to run time. *Negative sampling* outperforms *hierarchical softmax* in any respect; similar differences were observed in [29]. The handling of the *match count* in particular appears to be a critical parameter to obtain good models. Our results show that working with one billion words and ignoring the Ngram frequency yields models not very different from the reference model for data from the recent 50 years. The models trained on historical data before 1900 perform worse. We have seen that this is not only due to the number of questions the models cannot answer. Even when filtering the tests to only include questions that the models can answer, model performance in the 18th and 19th century is considerably lower than for the 20th century. One may think that a reason for this subpar performance is the volume of the data, and more data might help to stabilize the model training. The total corpus size before 1900–1904 does not reach one billion words. Additional tests however where we trained the models backwards, i.e., from 1850 to 1750, or by initializing with the reference model, have shown that the inferior performance is

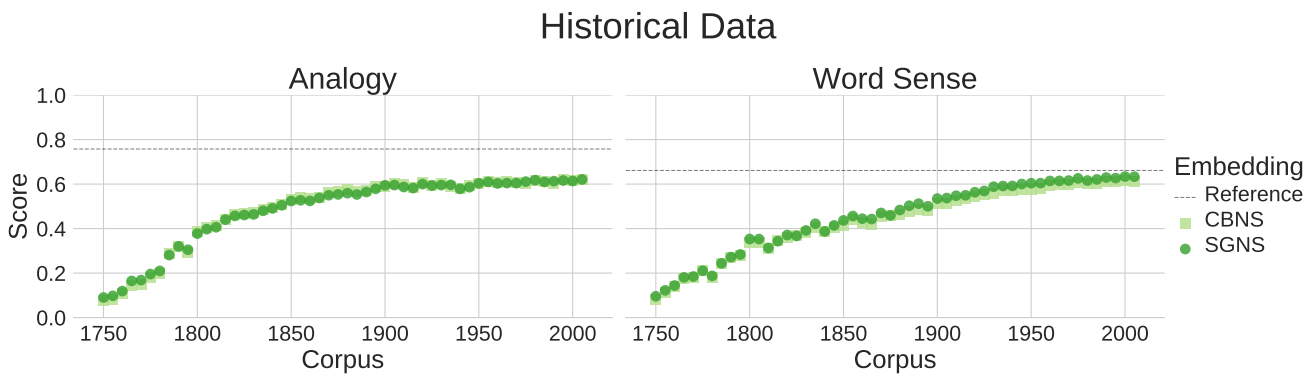

Fig. 6 Scores over the 5-year time periods of the Books data set.

not due to the volume of the data in the 18th and 19th century. We omit the presentation of detailed results, in order to not make the article too long. Instead the data itself is the problem when evaluated with tests for contemporary language. Whether this is because the language actually has changed so much, i.e., the language tests do not match the state of the historical corpora, or the data quality per se is worse, remains an open question. This would require expert knowledge of linguists and is beyond the scope of this current article. For our primary goal of analyzing semantic shifts, the extent of the evaluation is sufficient so far, as we have obtained important information on how to train language models from the Books Ngram data set.

5.2 Extrinsic Evaluation of Semantic Shifts

We now examine our time series model in the context of detecting semantic shifts. A problem here is that the ground truth of all shifts is unknown, and inspecting all detected changes manually is impossible. To evaluate different word embedding combinations, one needs an appropriate data set with a ground truth. Therefore, we follow the state-of-the-art approach in [5] and construct a synthetic corpus with manually induced semantic shifts. In order to prove the generality and validity of our approach, we seek answers to the following questions:

1. Micro-Benchmarking: Which combination of word embedding type, approximation algorithm, and alignment works best?
Rationale behind the question: By comparing different combinations, we prove the abstraction of our approach and at the same time identify the best setting for further analysis.

2. Inclusion of Frequency: How much does our model benefit from the frequency information included additionally?
Rationale behind the question: We seek to quantify the difference between executing our approach only on the semantic dimension against using both dimensions.

3. Comparison to state-of-the-art: Is our approach superior to previous work on semantic change analysis [4, 6, 25, 45] and detection [5]?
Rationale behind the question: To our knowledge, Kulkarni et al's work is the most recent work that performs change detection on time series built from word embedding models. We quantify the superiority of our approach and examine the limitations of [5]. Other methods for semantic shift analysis, such as [4, 6, 14, 25, 45], do not feature change detection. In other words, they do not return an estimate of the point in time when a word has shifted, only words with a high fluctuation. So while this is an easier problem, we use respective approaches as reference points nevertheless.

5.2.1 Synthetic Corpus

We now introduce the synthetic corpus. We follow the state-of-the-art approach in [5] for the construction. Our corpus is based on a full dump[4] of the English Wikipedia from November 1, 2016. After extracting all raw texts from the articles by removing all *xml*-meta tags and WikiMarkup[5], we model 10 time periods by duplicating the corpus 10 times. The last 5 time periods are then perturbed as follows: We select (*donor*, *receptor*) word pairs, where the *donor* is replaced with the *receptor* with a certain probability $p \in [0.1, 0.2, \ldots, 0.9]$, as in [5]. The *receptor* then is the word under investigation since it starts appearing in a new context, i.e., the context of the *donor* before the replacement. As word pairs, we manually choose 75 word antonym pairs, e.g., (small, big), (birth, death) or (begin, end). In order to

[4] The most recent dump is always available at `https://dumps.wikimedia.org/`

[5] `https://en.wikipedia.org/wiki/Help:Wiki_markup`

do an evaluation with different word types, the pairs consist of 20 adjective, 24 noun, 18 verb and 13 adverb pairs. The number of 75 pairs is small enough to not change the language too much and to allow a quantitative evaluation at the same time. Additionally, the selection of antonym pairs ensures that the grammar of a sentence is not completely destroyed. This is because most of the time antonyms can be exchanged. Antonyms sometimes appear in completely different contexts which should then result in heavy movement in the distributional language models. With this approach, we manually induce a semantic shift at time period 6. We expect change detection methods to then identify this point in time. After the perturbation, the text is transformed to 5grams and filtered exactly as the Google Books Ngram data set, to enable a realistic comparison. In order to realistically model different degrees of semantic shift, we use two types of perturbation:

- *Static* perturbation with a fixed $p \in [0.1, 0.2, \dots 0.9]$
- *Incremental* perturbation with temporally increasing value p from time period 6 to 10:
  slow: $0.1, 0.2, \dots$; *mid*: $0.1, 0.3, \dots$; *fast*: $0.3, 0.6, \dots$

This type of perturbation reflects the natural language where not all semantic shifts are sudden but sometimes also steady trends.

As an example, the time series for the word "birth" is in Figure 7. On the left side, the raw values are visualized similarly to our description in the *dimension combination problem* paragraph. On the right, there is the same with our model B; note the different y-axis. The time series is modeled from incrementally trained SGNS embeddings with perturbation mode *fast*. Clearly, the synthetic change at time period 6 is present, and relative frequency increases as well.

We note that it is only the frequency of the words affected by the perturbation that changes in this synthetic corpus, i.e., *donor* and *receptor*. In real world corpora this is not the case. This is because frequency also fluctuates without any semantic shift. A baseline detection solely run on the frequency dimension of our synthetic corpus performs well. However, it is not clear how one could induce such frequency fluctuations by hand without modifying the word sense. So we follow our main competitor [5] in the creation of the synthetic corpus. Our two-dimensional model still outperforms a baseline run on the frequency dimension, as we will see in Section 5.2.3.

5.2.2 Metric

We now discuss the metric used for this evaluation. We define the quality criterion as where shifted words appear in the ranked list after sorting by change detection score. For the synthetic evaluation, the ground truth is available. So we can define a score based on the positions of the perturbed words in the ranking. To this end, we choose the *Mean Reciprocal Rank (MRR)*, as in [5]. Function $\text{rank}(w)$ is the position of word w in a sorted list. Given the perturbed words $\mathcal{V}_\text{p}$ and the ranked list, we calculate:

$$\text{MRR} = \frac{1}{|\mathcal{V}_p|} \cdot \sum_{w \in \mathcal{V}_p} \frac{1}{\text{rank}(w)} \tag{4}$$

The larger MRR, the higher the quality is. Table 4 is an example where we calculate the MRR for two simple rankings: The inverse ranks of the words that actually have shifted are averaged.

Table 4 Illustration of the MRR calculation for two example rankings R_1 and R_2. The second and the fourth column state whether the word with the current rank has actually shifted with "yes"/"no". The total number of shifted words in this example is 3.

	R_1		R_2	
Rank	w shifted?	$1/\text{rank}(w)$	w shifted?	$1/\text{rank}(w)$
1	yes	$1/1$	no	
2	no		no	
3	yes	$1/3$	yes	$1/3$
4	yes	$1/4$	yes	$1/4$
5	no		yes	$1/5$
MRR		0.5278		0.2611

Generally, it is possible that several words have the same change detection score. A trivial detection that assigns a fixed-value score for all words results in a high MRR value. This is because all words would be placed on rank 1. To solve this issue, we penalize identical scores: We calculate unique ranks in the list and average the rank of groups with identical score. For example, a group of three words at the top of the ranking with identical score receives rank $(1 + 2 + 3) / 3 = 2$ for all three words. Additionally, to ensure that each perturbed word is detected in the time period when perturbation started, i.e., time period 6, we penalize wrong detections by setting the score to $\pm\infty$, depending on the value order of the change detection method used. As a consequence, a wrong detection appears at the end of the list, and the MRR score is reduced.

In opposition to other measures, the value range of MRR differs, and the perfect detection would be all 75 manually perturbed words appearing first in the ranked list. The corresponding score would then be 0.0654, which we add as a reference line to the plots. While we have only perturbed a few words, some other words can be affected as well, e.g., neighbors of the inserted word. This affects the final score since we cannot con-

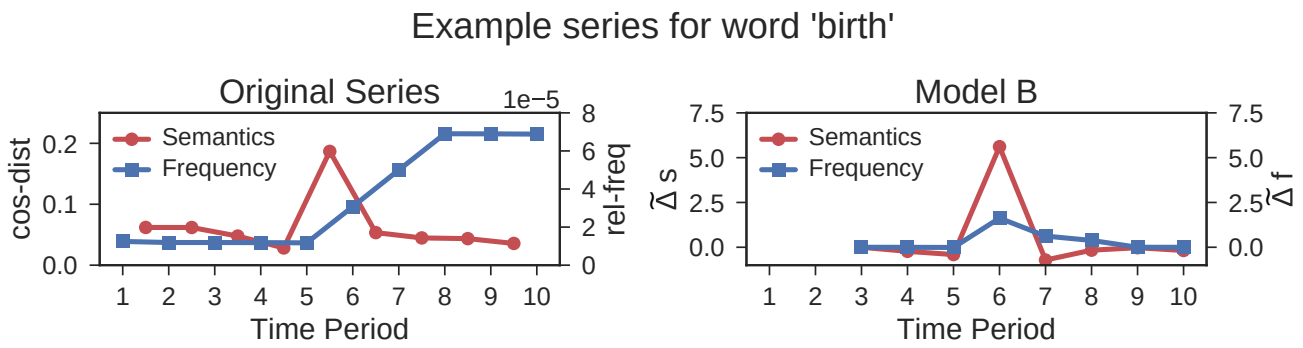

Fig. 7 Example time series for the word "birth" from incrementally trained SGNS embeddings.

trol the full impact of the perturbation. Nevertheless, our metric facilitates a quantitative comparison of different parameters. This fulfills our goal.

We use an additional metric to quantify whether a method detects the semantic shift at the correct point in time. We map the changes identified for the *receptor* words to a binary label. This label indicates whether the change is identified at the correct point in time, i.e., time point 6, or not. Then we use the true positive rate as a metric, i.e., the ratio of the correctly identified changes. Here, a value of 1 stands for a perfect detection, and 0 is the worst one. We refer to this metric as detection rate in the following.

5.2.3 Results

We now answer the questions raised earlier.

Micro-Benchmarking We test different combinations similarly to the previous evaluation. We seek the best combination for our remaining experiments. The results for different combinations for *static* and *incremental* perturbation are in Figure 8. The difference between NS and HS is again evident since NS consistently outperforms HS. Furthermore, our method identifies the correct change point for at least 95% for all perturbations with SGNS.

We now compare the alignment approaches. We evaluate models with SGNS that are trained incrementally, not aligned or aligned with Procrustes analysis or a linear transformation. To measure the raw impact of the alignment, we perform our detection only on the first dimension, i.e., the semantic movement. The results are in Figure 9. The incrementally trained models are superior to all other approaches. The alignment based on Procrustes analysis yields inferior results. It clearly outperforms the alternative without any or with the linear alignment though.

Inclusion of frequency As a next step, we quantify the MRR increase due to the inclusion of the frequency. Since we have extended the model with a second dimension for frequency, we can also run our detection only on the semantic movement dimension, in the following called 1D. In Figure 10 we compare the results of 1D and 2D detection for different combinations trained incrementally. The models with HS in particular close the gap to NS and highly benefit from the additional information. CBNS moves closer to SGNS as well and outperforms the CBNS 1D detection. For SGNS, detection improves only slightly but is still noticeable. A baseline that is computed by only running the change detection on the frequency dimension yields an average MRR of 0.0401. The detection on the frequency dimension is an easier task, due to the nature of the perturbations, as explained earlier. Yet our 2D detection outperforms this baseline as well with an average MRR of 0.04347. These results indicate that our abstraction works as intended and generally improves semantic shift detection.

Comparison to state-of-the-art We now compare with the model and the change detection method used in [5]. We also use incrementally trained series and follow the authors by computing the distances of each word to a fixed point instead of pairwise, as for our models. To ensure a fair comparison, we have tested multiple combinations including those used in their paper. Table 5 lists results from the best combination with 5000 random permutations, no threshold, and individually normalized time series. Our approach consistently outperforms detection based on the mean shift model. Regarding the detection rate, their approach only detects around 51% of the changes in the correct time period, compared to our 95%. Our CUSUM-based approach on the full vocabulary is several orders of magnitudes faster than their approach with 5000 permutations (around 6 hours for 1000, 32 hours for 5000 compared to our 10–15 seconds).

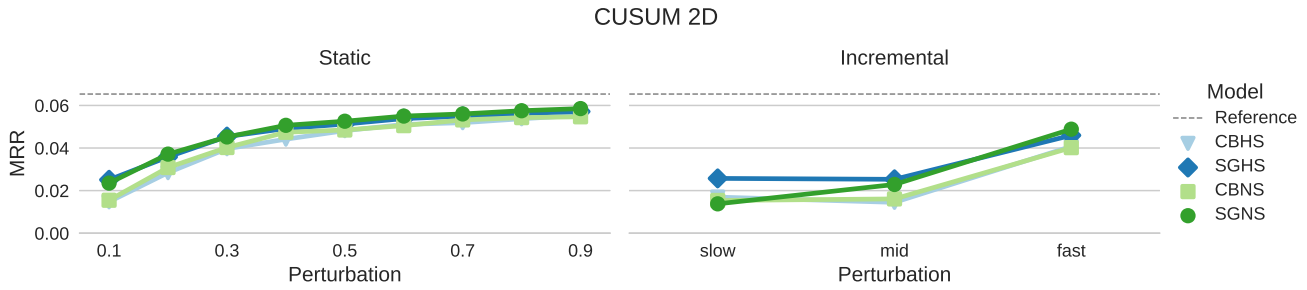

Fig. 8 Performance of different word embedding combinations trained incrementally.

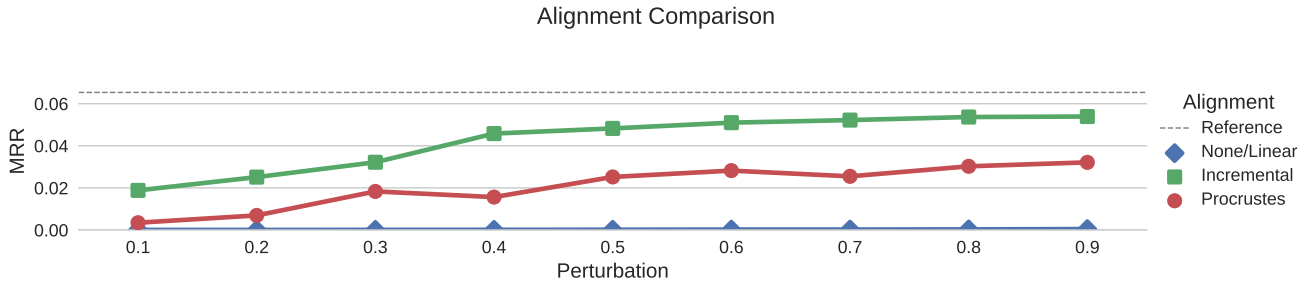

Fig. 9 Comparison of different alignment approaches using SGNS and CUSUM 1D on the cos-dist based dimension.

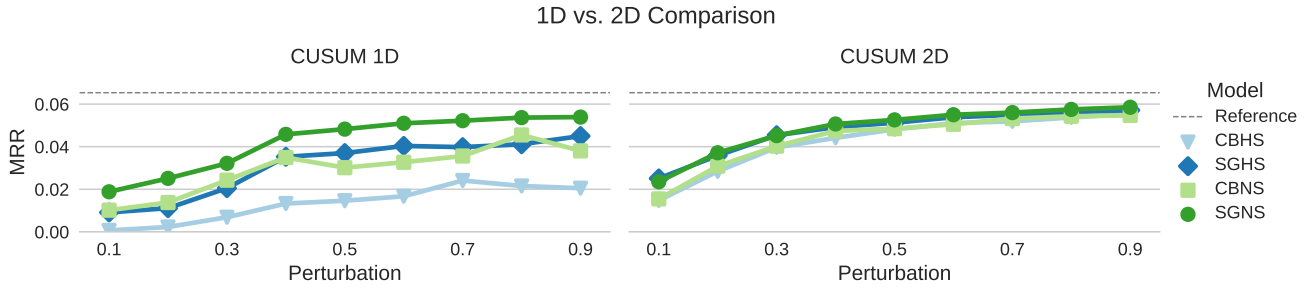

Fig. 10 Performance on only the similarity series (1D) against the extended model with frequency (2D).

We now compare against methods solving the easier problem of searching for words with the largest movement, without identifying a change point. To do so, we sum the cos-dist dimension as in [4, 6, 25, 45]. Table 5 lists the results. For SGNS and static perturbation, this sum based method performs worse than our method, with an average score of 0.0347 compared to 0.0485. In a more natural setup with incremental perturbation, SCAF outperforms the method relying on basic sums (average of 0.0285 against 0.0118).

5.2.4 Discussion

The first part of the evaluation has assessed different combinations of embedding types, approximation algorithms, and alignment methods. Generally, we suggest using SGNS trained incrementally. The disadvantage is that the word embedding models must be trained

Table 5 MMR score of the approach with change detection [5] and the one based on summing the semantic dimension [4, 6, 25, 45] on our synthetic corpora.

	[5]		[4, 6, 25, 45]†	
p	CBNS	SGNS	CBNS	SGNS
0.1	0.000048	0.000110	0.011134	0.007503
0.2	0.000027	0.000057	0.014936	0.014700
0.3	0.000032	0.000117	0.023319	0.023921
0.4	0.000030	0.000060	0.028694	0.029536
0.5	0.000038	0.000038	0.030801	0.035544
0.6	0.000035	0.000071	0.030560	0.048368
0.7	0.000028	0.000058	0.034111	0.049124
0.8	0.000031	0.000132	0.034777	0.052122
0.9	0.000037	0.000051	0.037270	0.051646
slow	0.000026	0.000107	0.009523	0.003622
mid	0.000026	0.000034	0.010977	0.003410
fast	0.000038	0.000054	0.022741	0.028464

†These methods do not identify any change point.

consecutively. When working with more time periods and a time constraint, we propose using individually trained models and aligning with Procrustes analysis. This allows to train models independently of each other in parallel.

Experiments have shown that the detection improves with our model. With the frequency dimension added, more information is available during change detection, and inferior combinations, e.g., with HS, profit in particular. Our method clearly outperforms the approach in [5]. Our average MRR score is 0.04347 compared to their score of 0.000074. For the detection rate, our method detects 95% of the changes at the correct point in time, compared to their 51%. The main problem of [5] is the calculated scores: Since the score directly depends on the number of permutations, the suggested number of 1000 produces equivalent scores for a lot of words, and increasing to 5000 permutations hardly helps. As stated, we compute the average rank in case of score equivalence. It is not clear how [5] deals with equivalent scores. As discussed when explaining the MRR, fixing this equivalence in the ranking is crucial for a meaningful result. Another strength of our CUSUM-based approach is the runtime, which is several orders of magnitudes lower than their approach. On the full vocabulary, our method runs in 10-15 seconds, compared to 6-32 hours of [5].

To sum up, we have shown that the method that sums up the semantic dimension without any change detection as in [4, 6, 25, 45] performs worse than our method. Some words are very unstable in the embeddings, i.e., cos-dist > 0.25. This includes very rare words as well as some high-frequency words for which the competitors cannot detect any semantic change, e.g., "separately" or "sensational". These words take the first ranks when summing the cos-dist values. Filtering such words on both ends is difficult. SCAF solves this problem by only summing changes of cos-dist values. We detect both rare as well as frequent words without being affected by the unstableness of the embeddings.

In this section, we have evaluated the different word-embedding-parameter combinations. Using a synthetic data set with a known ground truth, we have identified the best combination. With this knowledge we can now turn to the two real-world data sets.

5.3 Application to Books Corpus

In this section, we focus on the Books corpus. We first check for known shifts from related work. Then we proceed by identifying semantic changes of words that are unknown so far.

Table 6 Detection of CUSUM 1D and 2D on shifts known from related work [4, 5].

		CUSUM	
Word	Year	1D	2D
apple	1984	1980	1980
bitch	1955	1990	2005
bush	1989	1975	2005
diet	1970	1820	1975
gay	1920	1975	1995
honey	1930	1965	1945
monitor	1930	1960	1995
plastic	1950	1940	1975
record	1920	1810	1955
recording	1990	1915	1985
sex	1965	1940	1975
tape	1970	1900	1995
windows	1992	1985	2000

5.3.1 Known Shifts

We now test against known shifts:

Shift direction The first dimension refers to the movement of words in the vector space. While it is commonly used for semantic shift detection, only the raw similarities or distances are used while the vector direction is ignored, and the neighborhood is inspected for only a few time periods, e.g., in [5, 6]. To ensure that a word actually moves in the correct direction, we apply a test proposed in [4]. Based on word pairs, we evaluate whether the representative word vectors move towards or away from each other. One example is the word "gay" that started moving away from "happy" after 1920 and at the same time towards "homosexual". We calculate the results as in [4], by testing for the correct sign of the Spearman correlation between the pairwise cosine similarity and the time.

Results. With our incrementally trained models, we achieve an accuracy of 0.9643 for SGNS at a significance of 0.8929 at the $p < 0.05$ level. This test serves as a sanity check and confirms that our embeddings capture the correct shift directions.

Detection We now inspect the results on words that shifted according to related work [4, 5]. An excerpt of annotated shifts and our detection is in Table 6. The results are for the 1D detection as well as for the 2D detection. The 1D detection yields an average deviation of 47.69 years, compared to the 26 years for the 2D detection. As presented earlier, the embeddings from data in the 19th century are of inferior quality; this affects the 1D detection. For the words "diet" and "record", 1D detection falsely positions the changes at very early

Table 7 Top changes in time period 1900–2005 identified by our method. The explanations are from the etymology dictionary.

Word	Year	Explanation
mores	1950	From Latin "mores" meaning "manners" used since around 1900-1910
tipper	1990	Dump truck (since around 1910) and Mary Elizabeth "Tipper" Gore second lady 1993–2001
abusers	1980	Appeared in the context of drugs starting in 1960s–1970s
reparations	1925	Meaning "compensation for war damaged owed by the aggressor" attested from 1920s (referring to Germany)
childrens	1975	illegal plural form with increasing usage in broadlydifferent contexts

points in time. With the additional frequency information, this problem is mitigated.

Our method places the shifts later than the actual year, for several reasons: Semantic shifts often are not immediate, but a steady process. Additionally, we work with a book corpus where changes can appear later. During the shift, it also takes some time until the new word sense is generally adopted. This causes a slight delay in the word usage and therefore in occurrence in books as well. This explains the small temporal displacement, and our method places the changes at positions similar to the ones from related work.

5.3.2 Exploration

As a last step, we inspect the detection results of our method on the time period 1900–2008. Additionally, we filter the vocabulary with the English word list provided in [4], to get rid of stop words and proper nouns. First and last names dominate top lists of most changed words since they appear in continuously changing context. As noted earlier, words that appear as a proper name later are still under investigation. We also filter words appearing less than 1000 times. All in all, this results in a vocabulary of around 20,000 words.

Table 7 lists the top words detected by our method in 1900–2008 and gives an explanation. We identify the reasons by inspecting the nearest neighbors in the embedding models and by consulting an etymology dictionary[6]. We can find an explanation for most of the words in the first positions of the list. An inspection of the time series shows that all words at the top of the list moved in the embeddings while relative frequency increased at the same time.

Additionally, we explore the most recent years by running our model and change detection on the time period 1975–2008. Table 8 contains some of the words that changed most. For these, the shift is evident and can also be observed by inspecting the nearest neighbors. Terms with new meanings in the area of technology dominate the list. Their usage increases distinctly in the last years. As an example, we give the original series together with our time series model for the word "excel" in Figure 11. The semantic shift after 1985 is evident. While the cosine distance already drops again after 1990, relative frequency still increases heavily. The calculated cumulative sum therefore increases until 1995 before stabilizing. Our method then reports 1995 as the most probable change.

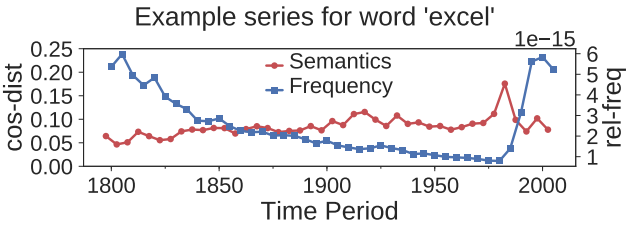

Fig. 11 Example time series for the word "excel" with the clear semantic shift after time period 1985.

5.3.3 Discussion

We have evaluated our approach on the Books data set. It can detect valid change points for words that knowingly shifted. It also has identified new semantic changes. Including the relative frequency in our time series model affects the change detection results. A core insight is that our approach identifies a new kind of word that has semantically shifted: Our method ranks words higher that not only have shifted semantically but have also gained popularity at the same time. The full detection list for different time spans is part of our supplementary material.

5.4 Application to Twitter data

In this section, we focus on the Twitter data set. Due to the lack of a ground truth on this very recent data, we perform an exploratory analysis. For this analysis we train SGNS models incrementally. We initialize the first

[6] `http://www.etymonline.com/`

Table 8 Some top words extracted in period 1975–2008.

Word	Year	Neighbors 1900	Neighbors 2005
server	1995	servers, player	servers, dhcp
web	1995	unweaves, webs	httpwww, namespaces
gender	1990	nouns, infinitives	sexes, heterosexual
navigate	2000	navigating, steer	navigating, upload
java	1990	sumatra, ceylon	servlets, applets
excel	1995	surpass, excelled	submenu, autoshape

model with the previously mentioned reference model from the *word2vec* toolkit.

5.4.1 Exploration

We now inspect the results SCAF produces for the Twitter data set. Table 9 lists the top words detected by our method. Award shows are hot topics which are heavily discussed on Twitter. Celebrities are another typical type of word that appears in changing context. Our method also is able to detect sudden hypes at the correct time. Examples are the "mannequinn challenge" or "Pokémon Go".

5.4.2 Discussion

We have applied our approach on a sample of the Twitter stream. It can detect reasonable change points for hot topics including award shows, celebrities or other trends. Semantic changes in the Twitter stream are more sudden than in the Google Books Ngram corpus. This is because social events immediately result in new Tweets while publishing books incurs some delay. This behavior is similar to the static perturbation setup of our synthetic data set which we have evaluated.

6 Conclusions

Recent research on automatic shift detection of word semantics has made significant advances relying on word embeddings. Web search in particular has to be aware of the evolution of language, in order to adapt to recent hypes or trends. However, a limitation of previous approaches is that the rate of semantic shift negatively correlates with frequency. To address this issue, we have investigated how to incorporate word usage frequencies. To this end, we developed SCAF, an abstraction of the time series models used in related work with an orthogonal extension by frequency. While we focused on different combinations of embedding type, approximation algorithm, and alignment method, the abstraction is independent of the underlying embeddings and hence allows applying any approach where the rate of semantic change is measurable. We thereby rely on the well-known change detection mechanism CUSUM. Based on that, we examine how to include word frequencies so that our model can be used with any combination. Our evaluation relies on the idea in [5] to create synthetic semantic shifts and achieves better detection rates than the main competitor [5]. On real-world data, our approach detects more plausible change points than previous work. It also detects yet unknown shifts for popular words.

Regarding the analysis of semantic shifts, SCAF opens a new field that incorporates the changing popularity of a word. With the published material, i.e., the word embedding models, we ease the work of linguists extracting new knowledge from the steadily evolving language.

Future work This article raises several open questions for future work. In Section 5.1 we have analyzed the performance of word embedding models trained on historical corpora. The poor performance in the 19th century has surprised us. Future work should assess the quality of the Books data set with help of expert knowledge of linguists. Additionally, to match historical language more suitable word embeddings tests should be developed. This would allow to evaluate the quality of embeddings trained on historical corpora.

In this article we also have studied semantic shifts on a word basis. Naturally, a lot of words have several meanings. With the evolution of language some appear or disappear. An interesting research direction would be to study the evolution for each meaning of a word individually. One can identify the different meanings by applying word-sense disambiguation as a preprocessing step. The different meanings could then be studied individually. However, it is not clear whether it is feasible to apply such a disambiguation on the corpora used in this article. This is because the disambiguation requires suitable resources, e.g., historical dictionaries in our case. This seems particularly difficult for the very recent Twitter data with its trends and hypes. Future research could address this problem of performing word-sense disambiguation on historical

Table 9 Excerpt of the top words that changed in January 2016 – June 2017 identified by our method on Twitter data.

Word	Date	Explanation
mtvhottest	July 2016	The hashtag is used for artist voting preceding the MTV Video Music Awards
amas	November 2016	Winners of the American Music Award appear with "amas"
camilacabello	March 2017	"mgk" (Machine Gun Kelly) appears in her context after their performance at the Kids' Choice Awards
mannequin	November 2016	The viral internet trend called "mannequin challenge" appears
comey	May 2017	Trump dismissed James Comey on May 9, 2017
merly/streep	January 2017	Both words appear for Merly Streep winning a honorary Golden Globe on January 8, 2017
magenta	May 2017	The color appears in the context of the T-Mobile Magenta Carpet Live event
phelps	June 2017	Swimmer Michael Phelps announces to race a great white shark
pokemon	July 2016	The augmented reality game Pokémon Go is released

corpora. When a respective method will have been developed in the future, our approach can then be applied to study semantic shifts further.

References

1. Zipf GK (2016) Human behavior and the principle of least effort: An introduction to human ecology. Ravenio Books
2. Schatz BR (1997) Information retrieval in digital libraries: Bringing search to the net. Science 275(5298):327–334
3. Michel JB, Shen YK, Aiden AP, Veres A, Gray MK, Pickett JP, Hoiberg D, Clancy D, Norvig P, Orwant J, et al (2011) Quantitative analysis of culture using millions of digitized books. science 331(6014):176–182
4. Hamilton WL, Leskovec J, Jurafsky D (2016) Diachronic word embeddings reveal statistical laws of semantic change. In: ACL, vol 1, pp 1489–1501
5. Kulkarni V, Al-Rfou R, Perozzi B, Skiena S (2015) Statistically significant detection of linguistic change. In: WWW, pp 625–635
6. Kim Y, Chiu YI, Hanaki K, Hegde D, Petrov S (2014) Temporal analysis of language through neural language models. In: ACL, pp 61–65
7. Jatowt A, Duh K (2014) A framework for analyzing semantic change of words across time. In: IJDL, pp 229–238
8. Basile P, Caputo A, Semeraro G (2015) Temporal random indexing: A system for analysing word meaning over time. Italian Journal of Computational Linguistics 1(1):55–68
9. Phillips L, Shaffer K, Arendt D, Hodas N, Volkova S (2017) Intrinsic and extrinsic evaluation of spatiotemporal text representations in twitter streams. In: Proceedings of the 2nd Workshop on Representation Learning for NLP, pp 201–210
10. Basile P, Caputo A, Semeraro G (2016) Temporal random indexing: a tool for analysing word meaning variations in news. In: ECIR, pp 39–41
11. Yao Z, Sun Y, Ding W, Rao N, Xiong H (2018) Dynamic word embeddings for evolving semantic discovery. In: WSDM, pp 673–681
12. Kendall DG (1953) Stochastic processes occurring in the theory of queues and their analysis by the method of the imbedded Markov chain. The Annals of Mathematical Statistics pp 338–354
13. Zhang Y, Jatowt A, Bhowmick SS, Tanaka K (2016) The past is not a foreign country: Detecting semantically similar terms across time. TKDE 28(10):2793–2807
14. Hamilton WL, Leskovec J, Jurafsky D (2016) Cultural shift or linguistic drift? comparing two computational measures of semantic change. In: EMNLP, pp 2116–2121
15. Basseville M, Nikiforov IV, Others (1993) Detection of abrupt changes: theory and application, vol 104. Prentice-Hall, Inc.
16. Taylor WA (2000) Change-point analysis: a powerful new tool for detecting changes
17. Ghanbarnejad F, Gerlach M, Miotto JM, Altmann EG (2014) Extracting information from s-curves of language change. Journal of The Royal Society Interface 11(101):20141044
18. Piantadosi ST (2014) Zipf's word frequency law in natural language: A critical review and future directions. Psychonomic bulletin & review 21(5):1112–1130
19. Krishnamoorthy N, Malkarnenkar G, Mooney R, Saenko K, Guadarrama S (2013) Generating Natural-Language Video Descriptions Using Text-

Mined Knowledge. In: AAAI, pp 541–547
20. Bassil Y, Alwani M (2012) Ocr post-processing error correction algorithm using google's online spelling suggestion. Journal of Emerging Trends in Computing and Information Sciences 3(1)
21. Nazar R, Renau I (2012) Google books n-gram corpus used as a grammar checker. In: EACL, pp 27–34
22. Mikolov T, Corrado G, Chen K, Dean J (2013) Efficient Estimation of Word Representations in Vector Space. ICLR
23. Pennington J, Socher R, Manning CD (2014) GloVe: Global Vectors for Word Representation. EMNLP pp 1532–1543
24. Mikolov T, Sutskever I, Chen K, Corrado G, Dean J (2013) Distributed Representations of Words and Phrases and their Compositionality. NIPS pp 3111–3119
25. Muromägi A, Sirts K, Laur S (2017) Linear ensembles of word embedding models. In: Proceedings of the 21st Nordic Conference on Computational Linguistics, pp 96–104
26. Rudolph M, Blei D (2017) Dynamic bernoulli embeddings for language evolution. arXiv preprint arXiv:170308052
27. Gladkova A, Drozd A (2016) Intrinsic evaluations of word embeddings: What can we do better? In: ACL, pp 36–42
28. Schnabel T, Labutov I, Mimno D, Joachims T (2015) Evaluation methods for unsupervised word embeddings pp 298–307
29. Hellrich J, Hahn U (2016) An assessment of experimental protocols for tracing changes in word semantics relative to accuracy and reliability. In: SIGHUM, pp 111–117
30. Levy O, Goldberg Y, Dagan I (2015) Improving Distributional Similarity with Lessons Learned from Word Embeddings. TACL 3:211–225
31. Elekes Á, Englhardt A, Schäler M, Böhm K (2018) Toward meaningful notions of similarity in nlp embedding models. IJDL pp 1–20
32. Elekes A, Englhardt A, Schäler M, Böhm K (2018) Resources to examine the quality of word embedding models trained on n-gram data pp 423–432
33. Blank A (1999) Why do new meanings occur? A cognitive typology of the motivations for lexical semantic change. Historical semantics and cognition 13:61–89
34. Traugott EC, Dasher RB (2001) Regularities in Semantic Change. Cambridge University Press
35. Hopper PJ, Traugott EC (2003) Grammaticalization. Cambridge University Press
36. Bréal M (1904) Essai de sémantique: (science des significations). Hachette
37. Ullmann S (1962) Semantics: an introduction to the science of meaning. Barnes & Noble
38. Traugott EC (1989) On the rise of epistemic meanings in English: An example of subjectification in semantic change. Language pp 31–55
39. Durie M, Ross M (1996) The comparative method reviewed: regularity and irregularity in language change. Oxford University Press
40. Lin Y, Michel JB, Aiden EL, Orwant J, Brockman W, Petrov S (2012) Syntactic annotations for the google books ngram corpus pp 169–174
41. Gulordava K, Baroni M (2011) A Distributional Similarity Approach to the Detection of Semantic Change in the Google Books Ngram Corpus. In: GEMS, pp 67–71
42. van Aggelen A, Hollink L, van Ossenbruggen J (2016) Combining Distributional Semantics and Structured Data to Study Lexical Change. In: EKAW, pp 40–49
43. Del Tredici M, Nissim M, Zaninello A (2016) Tracing metaphors in time through self-distance in vector spaces. arXiv preprint arXiv:161103279
44. Basile P, Caputo A, Luisi R, Semeraro G (2016) Diachronic Analysis of the Italian Language exploiting Google Ngram. CLiC-it pp 56–60
45. Takamura H, Nagata R, Kawasaki Y (2017) Analyzing semantic change in japanese loanwords. In: EACL, vol 1, pp 1195–1204
46. Tahmasebi N, Gossen G, Kanhabua N, Holzmann H, Risse T (2012) Neer: An unsupervised method for named entity evolution recognition. COLING pp 2553–2568
47. Rehurek R, Sojka P (2010) Software Framework for Topic Modelling with Large Corpora. In: LREC, pp 45–50