A Hierarchical Playscript Representation of Distributed Words for Effective Semantic Clustering and Search

Avi Bleiweiss(✉)

BShalem Research, Sunnyvale, USA
avibleiweiss@bshalem.onmicrosoft.com

Abstract. Semantic word embeddings have shown to cluster in space based on linguistic similarities that are quantifiably captured using simple vector algebra. Recently, methods for learning distributed word vectors have progressively empowered neural language models to compute compositional vector representations for phrases of variable length. However, they remain limited in expressing more generic relatedness between instances of a larger and non-uniform sized body-of-text. A recent study proposed a formulation that combines a word vector set of variable cardinality to represent a verse, with an iterative distance metric to evaluate similarity in pairs of non-conforming verse matrices. In this work, we expand on this sentence abstraction and apply it to a dialogue text passage that is prescribed in a playscript and uttered by an actor. In contrast to baselines characterized by a bag of features, our model preserves word order and is more sustainable in performing semantic matching at any of a dialogue, act, and play levels. To validate our framework for training word vectors, we analyzed the clustering of the complete play set of Shakespeare by exploring multidimensional scaling for visualization, and experimented with playscript searches of both contiguous and out-of-order parts of dialogues. We report robust results that support our intuition for measuring play-to-play and dialogue-to-play similarity.

1 Introduction

The attraction of using vector spaces for analyzing natural language semantics, stems primarily from providing an instinctive relation mechanism by subscribing to lexical distance and similarity concepts. In a large corpora of text, the structure of a semantic space is created by evaluating the various contexts in which words occur. Thus leading to distributional models of word meanings with the underlying assertion that words who transpire in similar contexts tend to have similar interpretations [26]. Distributed words, also known as word embeddings, are each represented with a dense, low-dimensional real-valued vector, and follow efficient similarity calculations directly from the known Vector Space Model [22]. Word vectors have been widely used as features in a diverse set of computational linguistic tasks, including information retrieval (IR) [17], parsing [24], named entity recognition [10], and question answering [12].

J. van den Herik et al. (Eds.): ICAART 2017, LNAI 10839, pp. 82–101, 2018.
https://doi.org/10.1007/978-3-319-93581-2_5

In recent years, neural network architectures have inspired the deep learning of word embeddings from large vocabularies to avoid a manual labor-intensive design of features. The work by Bengio *et al.* [4] introduced a statistical language model formulated by the conditional probability of the next word given all its preceding words in a sequence, or a context. However, the time complexity of the neural model renders the scheme inefficient due to the non-linear hidden layer. The succeeding Word2Vec [18] and GloVe [20] methods preserve the probabilistic hypotheses founded in Bengio *et al.* [4] approach, and further eliminate the hyperbolic tangent layer entirely, thus becoming more effective and feasible tools for language analysis. Notably, these methods expand on the input context window to include the words that both precede and follow the target word. Word embeddings are typically constructed by way of minimizing the distance between words of similar contexts, and encode various simple lexical relations as offsets in vector space. Our work investigates the linguistic structure in raw text, and explores clustering and search tasks using Word2Vec to train the underlying word representations.

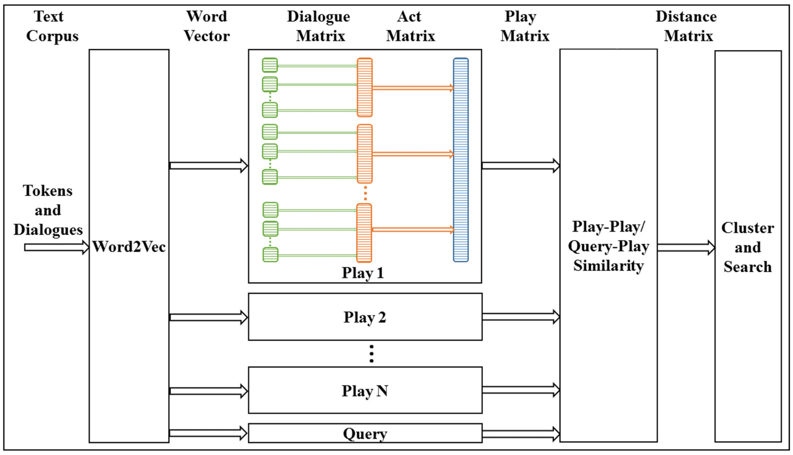


Fig. 1. Framework overview: on the left, tokens and context from a text corpus are used to train word vectors. A collection of word vectors is constructed to represent word-for-word the source text of every playscript. Word vectors are first row bound into a matrix to represent an actor specific or simultaneous dialogue. Then, dialogue matrices are concatenated into act matrices that are further coalesced into a hierarchical play matrix. We run all-play-pairs and all-query-play-pairs similarity process on matrices of a non-uniform row count, and generate a distance matrix that we use for cluster analysis and search ranking, respectively.

Applying unsupervised learning [8] of distributed word embeddings to a broader set of semantic tasks has not yet been fully established and remains an active research to date. In their recent work, Fu *et al.* (2014) utilize word embeddings to discover hypernym-hyponym type of linguistic relations. Noting that offsets of word pairs distribute into structured clusters, they modeled fine-grained relations by estimating projection matrices that map words to their respective hypernyms, and report a reasonable test F1-score of 73.74%. Socher *et al.* (2013) proposed a recursive neural network to compute distributed vector representations for phrases and sentences of variable length. Their model outperforms state-of-the-art baselines on both sentiment classification and accuracy metrics, however, its supervised method requires extensive manual labeling and makes scaling to larger sized text non trivial. A representation more rooted in a convolutional neural architecture [15], produces a feature map for each possible window in a sentence, and follows with a max-over-time pooling [6] to capture the most important features. Pooling has the apparent benefit of naturally adapting to variable length sentences. At the higher document level, Le and Mikolov [16] introduced a paragraph vector extension to the learning framework of word vectors. Given their different dimensionality, paragraph and word vectors are concatenated to yield fixed sized features, however, unique paragraph vectors constrain context sharing across the document.

For a composition of words, most of the techniques discussed tend to reshape the varying dimensionality of input sentences into uniform feature vectors. Rather, our implementation retains the word vectors as distinct rows in a matrix form to construct any of the dialogue, act, or play data structures for performing our set of linguistic tasks over a collection of scripts. The main contribution of our work is a framework (Fig. 1) with a lexical representation that abides word-for-word by the corpus source sequence, to facilitate a generic evaluation of relationships among entities of non-uniform text size. This work extends a recent study [5] by issuing responses to keyphrase queries that not only identify the act and scene enumerations inside a script, but in addition single out the corresponding actor names paired with the enclosed dialogue text-sequence. The rest of this paper is organized as follows. In Sect. 2, we briefly review the neural models to found Word2Vec, and proceed with motivating our choice for a dialogue matrix representation that leads to our act and play hierarchies. Section 3 derives our play similarity measure as it applies to a pair of non-conforming concatenations of dialogue embeddings, whereas Sect. 4 profiles the compiled format of the Shakespeare play-set corpus we used for evaluation. We then present our methodology for analyzing clusters of Shakespeare genres and ranking playscript searches of unsolicited keywords, and report extensive quantitative results of our experiments, in Sect. 5. We conclude with a discussion and future prospect remarks in Sect. 6.

2 Embedding Hierarchy

In Word2Vec, Mikolov *et al.* (2013a) proposed a shallow neural-network structure for learning useful word embeddings to support predictions within a local bi-directional context-window. Both the skip-gram and continuous bag-of-words (CBOW) models offer a simple single-layer architecture based on the inner product between a pair of word vectors. In the skip-gram version the objective is to predict the not necessarily immediate context words given the target word, and conversely, CBOW estimates the target word based on its neighboring context. As a context window scans over the corpus, the models attempt to maximize the log probability of the generated objective function, based on their respective multiple and single output layers, and training word vectors proceeds in a stochastic manner using back propagation. To improve upon accuracy and training time, Mikolov *et al.* (2013b) introduced both randomly discarding of frequent words that exceed a prescribed count threshold, and the concept of negative sampling that measures how well a word pairs with its context drawn from a noise distribution of a small sample of tokens. Empirically, neural model performance shown mainly governed by tunable parameters, including the word vector dimension, d, the symmetric context-window size, c_w, and the number of negative samples, s_n. Overall, skip-gram works well with a small amount of training data, while CBOW is several times faster to train.

The corpus we used for our study comprises a set of tens of playscripts and to train word embeddings, we first flattened the entire corpus into a linear array of dialogue text passages. We then proceeded to construct our basic data structures that culminate in an effective hierarchical representation of a play object, which is perceived nameless across subsequent clustering and search analyses. Let $w^{(k)} \in \mathbb{R}^d$ be the d-dimensional word vector corresponding to the k-th word in a dialogue. A dialogue text sequence S of length n is represented as a matrix

$$S = w^{(1)} \oplus w^{(2)} \oplus \ldots \oplus w^{(n)}, \tag{1}$$

where $\oplus$ is a row-wise binding operator and $S \in \mathbb{R}^{n \times d}$. S is thus regarded as a vector set and rows of S are considered atomic. To index a word vector in a dialogue, we use the notation s_k. Similarly, a play act A of m dialogues becomes a concatenation of dialogue matrices, $A = S^{(1)} \oplus S^{(2)} \ldots \oplus S^{(m)}$, where $A \in \mathbb{R}^{r_a \times d}$ and $r_a = \sum_{j=1}^{m} |S^{(j)}|$, and a play P comprises l act matrices, $P = A^{(1)} \oplus A^{(2)} \ldots \oplus A^{(l)}$, where $P \in \mathbb{R}^{r_p \times d}$ and $r_p = \sum_{i=1}^{l} r_a^{(i)}$. Respectively, a_j itemizes a dialogue matrix in an act, and p_i enumerates an act matrix in a play. Equation 2 provides an alternate matrix notation for each a dialogue, act, and play constructs.

$$S = \begin{bmatrix} w^{(1)} \\ w^{(2)} \\ \vdots \\ w^{(n)} \end{bmatrix} \qquad A = \begin{bmatrix} S^{(1)} \\ S^{(2)} \\ \vdots \\ S^{(m)} \end{bmatrix} \qquad P = \begin{bmatrix} A^{(1)} \\ A^{(2)} \\ \vdots \\ A^{(l)} \end{bmatrix}. \tag{2}$$

The length of a dialogue, $|S^{(j)}|$, and the number of dialogues per act, $|A^{(i)}|$, are varying parameters that we track and keep in a play map for traversing the play hierarchy. For play similarity computations, we often iterate a play matrix and access the entire collection of word vectors. Conveniently, we use a three dimensional indexing scheme, p_{ijk}, where each of i, j, and k points to an act matrix, dialogue matrix, and word vector, respectively. Space complexity for play embeddings is linear, $\mathcal{O}(lmn)$, and for a vocabulary that permits storing a 16-bit token enumeration instead, memory area required is thus reduced by a half. We further denote the corpus play set $T = \{P^{(1)}, P^{(2)}, \dots, P^{(N)}\}$, where N, or cardinality $|T|$, is the number of plays.

3 Play Similarity

Measuring similarity and relatedness between distributed terms is an important problem in lexical semantics [1]. Recently, the process of learning word embeddings transpired compelling linguistic regularities by simply probing the linear difference between pairs of word vectors. This evaluation scheme exposes relations that are adequately distributed in a multi-clustering representation [9]. However, a single offset term is insufficient to assess similarity between a pair of plays represented by non-conforming matrices, each potentially retaining many thousands of word vectors. For evaluating semantic closeness of a pair of plays, $p^{(u)}$ and $p^{(v)}$, we explored a similarity concept that expands on the Chebychev matrix distance [5] and is defined by

$$\mathrm{d}(u,v) = \frac{1}{|p^{(u)}|} \sum_{xyz} \left\{ \max_{ijk} \left(\mathrm{sim}(p^{(u)}_{xyz}, p^{(v)}_{ijk}) \right) \right\}, \tag{3}$$

where $|p^{(u)}|$ is the play cardinality that amounts to the total number of distributed word vectors for representing the play, and $|p^{(u)}| \neq |p^{(v)}|$. Whereas sim() is a similarity function that operates on two word vectors and takes either a Euclidean or an angle form. We chose cosine similarity [2] that performs an inner product on a pair of normalized vectors g and h, $\frac{g \cdot h^T}{\|g\|_2 \|h\|_2}$, and returns a scalar value as a measure of proximity. The time complexity of the distance algorithm is roughly quadratic, as for each word vector in play $p^{(u)}$, we find the closest word vector in play $p^{(v)}$, and then calculate the mean of all the closest distances derived formerly.

In our system, all possible pairs of the corpus playscripts, T, are evaluated for similarity in the context of a $|T|$-dimensional squared distance-matrix, D. Elements of the distance matrix are considered directional and hence imply $\mathrm{d}(u,v) \neq \mathrm{d}(v,u)$. Matrix D facilitates unsupervised learning for clustering plays that apart from knowing their individual representations, they are perceived as a collection of unlabeled objects. Following an identical vein, as the distance metric provided by Eq. 3 is generic and assumes opaque matrix pairs, our framework naturally extends the semantic distance intuition to express a query-play type of relations for conducting a search. A query, q, comprises an unsolicited keyphrase and as such abides by our dialogue matrix formulation, S. The distance $\mathrm{d}(q,u)$, where $u \in \{1, 2, \dots, |T|\}$, thus conveys a distinct relevancy measure for ranking the search of a query in each of the corpus plays contained in collection T. Our system reports back search results as a table of sorted distances paired with the play enumeration [7].

4 Shakespeare Corpus

We evaluated the performance of our model on playscripts written by William Shakespeare that we acquired online from the publicly available repository provided by Project Gutenberg [23]. The project cites the complete work of Shakespeare and comprises both plays and poems. Among the editions offered, we chose the plain-text eBook version that is distributed in a single file with instructive header separators between plays. Plays written by William Shakespeare are often divided into three major genres and include comedy, history, and tragedy type of content, as Table 1 lists the play names associated with each of the genre partitions. Our study pertains to mining text of a traditional playscript format, and thus excludes poetry type compositions by Shakespeare, such as the renown collection of 154 sonnets.

Table 1. List of Shakespeare play names classified by their respective genres.

Comedy	History	Tragedy
All's Well That Ends Well	Henry IV, part 1	Antony and Cleopatra
As You Like It	Henry IV, part 2	Coriolanus
The Comedy of Errors	Henry V	Hamlet
Cymbeline	Henry VI, part 1	Julius Caesar
Love's Labours Lost	Henry VI, part 2	King Lear
Measure for Measure	Henry VI, part 3	Macbeth
The Merry Wives of Windsor	Henry VIII	Othello
The Merchant of Venice	King John	Romeo and Juliet
A Midsummer Night's Dream	Richard II	Timon of Athens
Much Ado About Nothing	Richard III	Titus Andronicus
Pericles, Prince of Tyre		
Taming of the Shrew		
The Tempest		
Troilus and Cressida		
Twelfth Night		
Two Gentlemen of Verona		
Winter's Tale		

The Shakespeare dataset consists of 37 full-length titles, as 17, 10, and 10 plays subscribe to comedy, history, and tragedy genre groups, respectively. Every Shakespeare playscript contains a section header labeled dramatis personae, persons of the drama, that lists the main characters of the play. However, this list is often incomplete, as there are many background and offstage roles that emerge as the storytelling evolves. For our work, we have scanned each playscript from start to end and manually extracted a complete list of all the names of active actors in

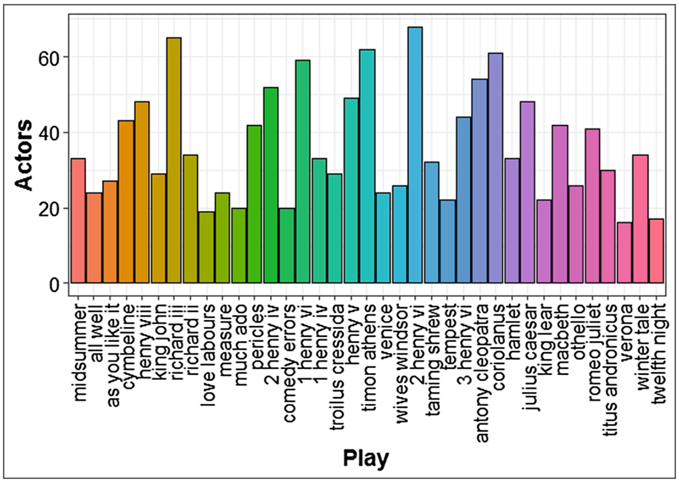


Fig. 2. Actor distribution across the entire Shakespeare play suite.

a play, and used it as a search index for binding a dialogue to an actor. In total, the Shakespeare corpus incorporates 1,352 distinct characters and Fig. 2 provides visualization of actor distribution across the entire play suite. Uniformly, every Shakespeare play is divided into five acts, as each act is composed of one or more scenes. For a scene, the playscript contains a brief header description of location and time of day, and may follow by appearance directions such as characters entering and exiting the stage. Successively, the script presents a chronology of dialouge text passages, each identified with a single or group of characters. A dialogue is often a short text sequence, but may also comprise a paragraph of several sentences. The Shakespeare corpus combines a total of 749 scenes and 30,473 dialogues, and visualization of per-act stacked distribution of scenes and dialogues across all the plays is outlined in Figs. 3 and 4, respectively. Table 2 provides complementary summarizations of actors-per-play, and aggregations over acts for scenes-per-play, and dialogues-per-play.

To derive our word vectors, we first tokenized and lowercased the dialogue text passages of the entire play suite, removed all punctuation marks excluding the hyphen, and have retained stop words of Old English that dates back to the Shakespeare era of the 17^{th} century. The corpus has a little under two million unfiltered symbols with a vocabulary of 24,747 unique tokens to train (Table 2). Notably, most unsolicited tokens are of a low frequency term with only 22 symbols exceeding 100 incidents. The mean frequency term is 56, and there are 196 tokens, under one percentage point of the vocabulary size, that occur only once in the dataset. In Fig. 5, we show our word cloud rendition using R [21] that depicts the top 100 frequent tokens in a linear dialogue collection extracted from the Shakespeare playscript suit. As anticipated, words of relational connotations like 'thou', 'thy', and 'thee', or of a romantic association such as 'good' and 'love', and monarchy related 'lord' and 'king' are of the highest occurrence term.

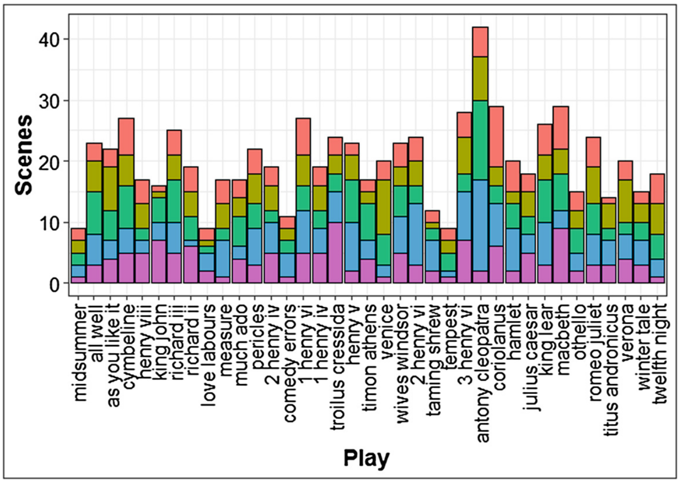


Fig. 3. Per-act stacked scene distribution across the entire Shakespeare play suite.

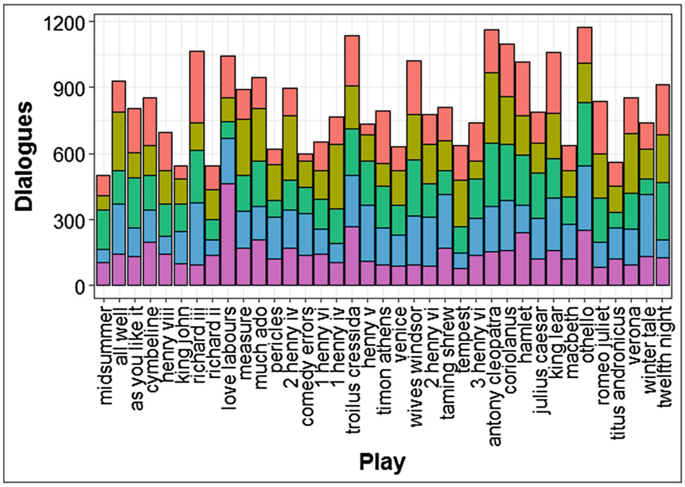


Fig. 4. Per-act stacked dialogue distribution across the entire Shakespeare play suite.

Dialogue text passages are regularly preceded in the playscript with a pattern consisting of an uppercased character name that is followed by a period. We note that this pattern remains verbatim and excluded from tokenization, and internally we retain a dialogue data structure as a record pair of the actor name and his or her corresponding text sequence. This lets us at any time respond to search queries with both a playscript location that comprises the act and scene IDs, and in addition provide the respective actor name. In addition, dialogues paired with multiple actors are fairly regular in Shakespeare plascripts, including the common appearances of LORDS, SERVANTS, ALL, and CHORUS, and both our actor index and representation abide by this simultaneous rule.

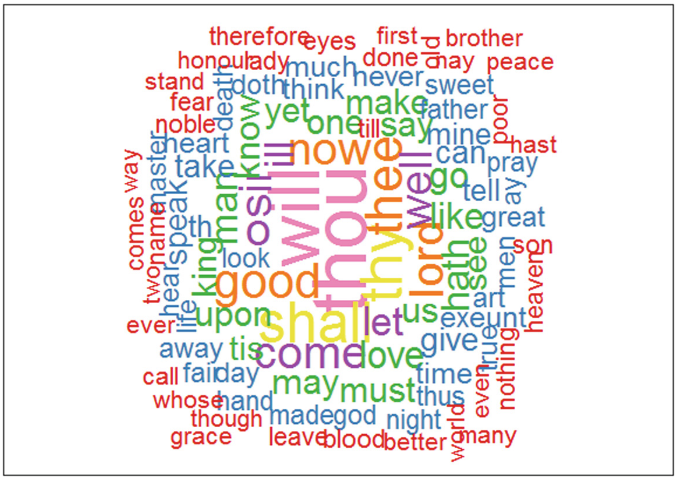


Fig. 5. caption A word cloud rendition of the top 100 frequent tokens as they appear in a linear dialogue collection extracted from the full set of Shakespeare playscripts. Font size is proportional to the number of word occurrences in the corpus.

To construct a context window, we randomly select an unlabeled play enumeration in the [1–37] range, and traverse our hierarchy top-to-bottom by arbitrarily sampling act and dialogue indices that are confined to the limits set by the singled out playscript. In our implementation, running dialogue indices cross scene boundaries and are presumed continuous throughout a given play act. In the text passage of the chosen dialogue, a random target-word position is used to extract from left and right context words that are delimited by the dialogue start and end words. Ultimately, one of our system goals is to discover semantic relations that closely align learned play clusters with the manually-prescribed classified genres listed in Table 1.

5 Empirical Evaluation

Previously, we discussed vector embedding techniques, such as Word2Vec [18] and GloVe [20], and their role to transform natural language words into a semantic vector space. In this section, we proceed to quantitatively evaluate the intrinsic quality of the produced set of latent vector representations, and analyze their impact on the performance of our unsupervised learning tasks that comprise play clustering and search. As an aid to tune our system level performance, we explored varying some of the hyperparameters designed to control the neural models incorporated in the word embedding methods. In practice, we have implemented our own Word2Vec technique natively in R [21] for better integration with our software framework. Across all of our experiments, we trained word vectors employing mini-batch stochastic gradient descent (SGD) with an annealed learning rate, and semantic similarity results we report on both play-to-play and dialogue-to-play relations presume anonymous plays.

Table 2. Summarizations of play properties including actors-per-play, aggregations over acts for scenes-per-play and dialogues-per-play, and at a corpus level the total of unsolicited and unique tokens.

Actors			
Min	Max	Mean	Total
16	168	36.5	1,352

Scenes			
Min	Max	Mean	Total
9	42	20.2	749

Dialogues			
Min	Max	Mean	Total
499	1,175	823.6	30,473

Tokens	
Unsolicited	Unique
1,999,034	24,747

5.1 Experimental Setup

The raw text of Shakespeare playscripts [23] underwent a data cleanup preprocess to decompose hyphenated word compounds, properly uppercase all instances of actor declarations, and introduce a more definite separation symbol between an actor ID and the succeeding dialogue text-sequence. We then tokenized the corpus using the R tokenizer [21] and built a maximal vocabulary V of size $|V| = 24,747$. Each word in this sparse 1-of-$|V|$ encoding is represented as a one-hot vector $\in \mathbb{R}^{|V| \times 1}$, with all its components zeroed out and a single one set at the index of the word in the vocabulary, that is further mapped onto a lower-dimensional semantic vector-space. Projecting onto the denser formulation transpires at the stage immediately preceding the hidden layer of the neural models that underlie the embedding technique.

In the absence of a large supervised training set of word vectors, the use of pre-trained word vectors obtained from an unsupervised neural model became a favorable practice to boost system performance [6,12,15]. While proven useful for word analogy and multi-class classification tasks, clustering and search over a rather unique dataset requires however randomly initialized word-vectors. Hence our model generates distinct input and output sets of word vectors, W and $\tilde{W}$, that only differ as a result of their random initialization. To help reduce overfitting and noise, we used their sum, $W + \tilde{W}$, as our final vectors and that

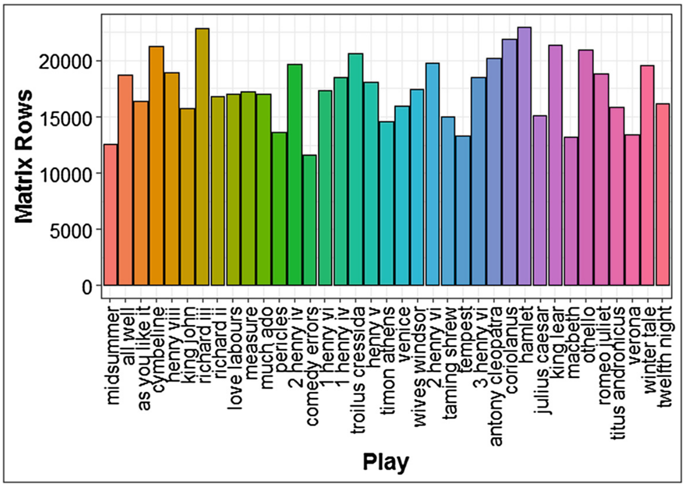


Fig. 6. Flattened play hierarchy into a single linear matrix of word vectors $w^{(k)} \in \mathbb{R}^d$. Showing distribution of matrix row count across the entire Shakespeare play suite.

typically yields a small performance gain. To better assess the space complexity of our play representation made of the trained word embeddings, Fig. 6 provides a visual interpretation of a flattened play hierarchy, outlined as a single linear matrix with up to several tens-of-thousands rows of word vectors, and shown distributed across the entire Shakespeare play suite.

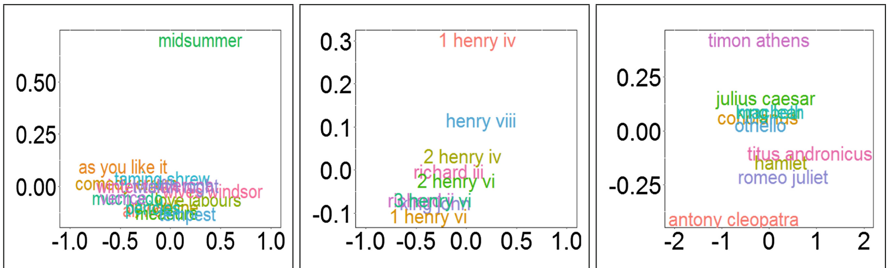
Recent study by Baroni *et al.* [3] alluded to neural word-embedding models that consistently outperform the more traditional count-based distributional methods on many semantic matching tasks and by a fair margin. Furthermore, much of the achieved performance gains cited are mostly attributed to a system design choice of the configured hyperparameters. Motivated by these results, we evaluated task performance comparing distinct play hierarchies generated by each skip-gram and CBOW, and choose negative sampling that typically works better than hierarchical softmax [19]. For the learning hyperparameters, there seems no single selection for an optimal word-vector dimension, d, as it tends to be highly task dependent and varies from 25 for semantic classification [24] up to 300 for word analogy [18]. Rather, we set $d = 10$ and traded-off word vector dimension to attain more tractable computation in building the distance matrix that is inherently of a quadratic time complexity, $\mathcal{O}(|T|^2)$. Whereas, to better assess the impact of the context window on our system performance, we varied discretely its size $c_w = \{5, 15, 25, 50\}$, in a wide range of word counts. Evidently, Word2Vec performance tends to decrease as the number of negative samples increases beyond about ten [20], thereby we used $s_n = 10$. For our mini-batch SGD to train word vectors, we used a batch size of 25 and an initial learning rate $\alpha = 0.1$, and updated parameters after each window. The number of epochs we ran in our experiments topped at 10,000, and is determined by both the chosen neural model and context window size, c_w.

5.2 Experimental Results

We present play clustering results of our own trained Shakespeare corpus at both the genre component level and for the entire suite of $|T| = 37$ plays. To visualize genre based clusters, we used Multidimensional Scaling (MDS) [11,25] that extracts patterns of semantic proximities from our play distance-matrix representation, D. The distance matrix is composed of a set of pairwise play-similarity values with $\mathcal{O}(|T|^2)$ scaling that MDS further compiles and projects onto an embedding p-dimensional Euclidean-space. This mapping is intended to faithfully preserve the clustering structure of the original distance data-points, and often, data visualization quality of clusters is directly proportional to the ratio $\frac{p}{|T|}$. In our experiments, we consistently rendered similarity measures of play pairs onto a two-dimensional coordinate frame to inspect and analyze formations of genre-based play clustering.

In Fig. 7, we provide baseline visualization of MDS applied to our play distance matrices that represent each of the genre components of comedy, history, and tragedy with dimensionality $|T| = \{17, 10, 10\}$, respectively. For this experiment, we used the CBOW neural model to train word vectors, as hyperparameters were uniformly set to their defaults, using 5 words for the context window size, c_w. The comedy collection shows most of its plays semantically closely related, with the exception of A Midsummer Night's Dream as an outlier with a fair distance apart. Whereas the history genre class of ten play samples has eight of its members arranged as immediate neighbors, the position of Henry VIII slightly disjoint, and the first part of Henry IV noticeably apart. On the other hand, the tragedy genre selection formed two distinct adjacent clusters of five and three plays, respectively, as the titles Antony and Cleopatra and Timon of Athens are notably detached.

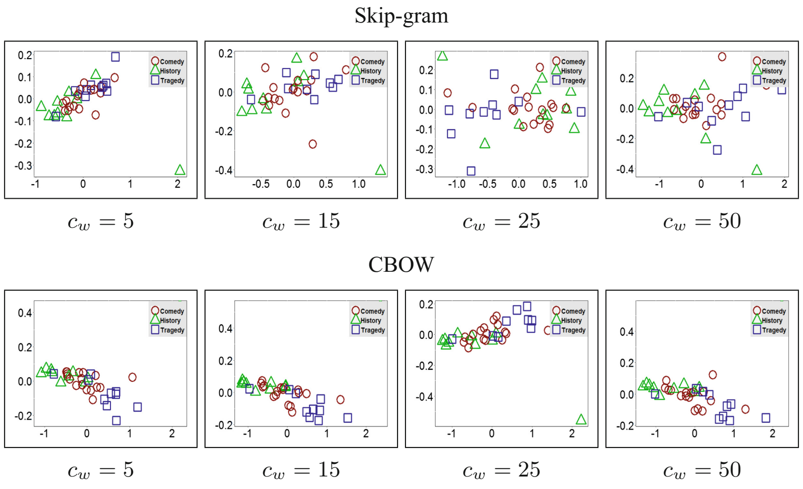
A much broader interest of our work underscores the cluster analysis of a single distance matrix with dimensionality $|T| = 37$ that represents the entire Shakespeare play suite. Through this evaluation, our main objective is to predict unsupervised play clustering and assess its matching to the Shakespeare formal



(a) Comedy. (b) History. (c) Tragedy.

Fig. 7. Visualization of play distance matrices using Multidimensional Scaling. Representing the Shakespeare genres of comedy, history, and tragedy each with 17, 10, and 10 plays, respectively.

Table 3. Visualization of clustering the entire Shakespeare play suite using Multidimensional Scaling, as each play is assigned its formal genre-subset legend post projecting onto the displayable embedding space. Shown as a function of a non-descending context window-size, c_w, for both the skip-gram and CBOW neural models.



genre subdivisions (Table 1). Table 3 shows the clustering produced by applying MDS to the single matrix that captures all-play-pairs semantic similarities, as each play is assigned its formal genre-subset legend post projecting onto the displayable embedding space. In these experiments, we compared unique word-vector sets generated by each skip-gram and CBOW, as we stepped over the fairly large extent of discrete values prescribed for the context window size, c_w. Expanding the context window scope has a rather mild impact on cluster construction with word vectors trained by the CBOW neural model, noting that for $c_w = 25$, clusters are shown shifted upwards as the mapping onto the history genre gave rise to one outlier, the Henry V play, at the bottom right corner of the plot area. However for skip-gram, group formations are considerably susceptible and affected by even a moderate change of c_w. Furthermore, the play partitions we generated under CBOW training persistently resemble the official subdivision of Shakespeare genre collections, although for visualization in a 2D embedding space, the comedy and history sets do overlap each other. The impact of the neural model used on clustering playscripts largely concurs with the results obtained using an orthogonal bible book-suite [5].

In our play search experiments, we explored three types of keyphrase queries including fixed dialogue text-passages drawn from a known play and act, reordered words of random partial dialogues distributed uniformly in each of the plays of the Shakespeare suite, and randomly selected tokens from the corpus vocabulary composed into a set of keywords. Every search is preceded by converting the query composition into a matrix of word vectors and then pairing the query matrix with each of the play hierarchies to compute similarity distances. Thus resulting in a process of linear time complexity, $\mathcal{O}(|T|)$. Unless otherwise noted, for the search experiments we applied default hyperparameters to CBOW-based trained word-vectors.

Table 4. Search queries of fixed dialogue text-sequences from known play origins.

Play	Actor	Search Query
A Midsummer Night's Dream	HIPPOLYTA	*bent in heaven shall behold the night*
Much Ado About Nothing	HERO	*says the prince and my new trothed lord*
King John	BASTARD	*bragging horror so shall inferior eyes*
Hamlet	QUEEN	*did he receive you well*
Macbeth	MACBETH	*our will became the servant to defect*

In Table 4, we list search queries of fixed dialogue text-passages and correspondingly enumerate their play origins and bound actors. Overall, for each of the five dialogues searched the predicted play title matched the expected label and was ranked highest with a score of 1.0. King John appeared to be a single source play for its keyphrase as the rest of the plays scored fairly low with a mean of 0.27. However, not surprisingly the search of the remaining four queries uncovered additional unlisted plays that equally claimed a lead score, as keyphrase words either extend within an act or over acts non-adjacently and would still rank high for relevance in the context of a play search. For instance, the Twelfth Night play scored high on the keyphrase from the play of Hamlet with keywords split apart among the first three acts, and similarly Cymbeline ranked high for the dialogue text passage from Macbeth, as keyword groups appear in each of the first and fifth acts. Figure 8 provides visualization to our fixed-dialogue search results in the form of a depleted confusion matrix $\in \mathbb{R}^{37\times5}$, with actual and predicted plays listed at the bottom and to the left of the grid, respectively.

In the second search experiment, we selected from each of the Shakespeare plays five random samples of partial dialogues, each with eight unordered context words. We ran a total of $5 \times 37 = 185$ search episodes and constructed a search matrix by computing all directional pairs of query-play distances, and then averaged the score for multiple queries per play. In Fig. 9, we show our results for the random sub-dialogue search and demonstrate consistent top ranking for when the source play of the queries matches the predicted play along the diagonal of the fully populated confusion-matrix $\in \mathbb{R}^{37\times37}$.

Our third task deployed cumulatively a total of 200 searches using a query keyphrase that is a composite of randomly selected tokens from our entire vocabulary, and thus implies a weak contextual relation to any of the Shakespeare

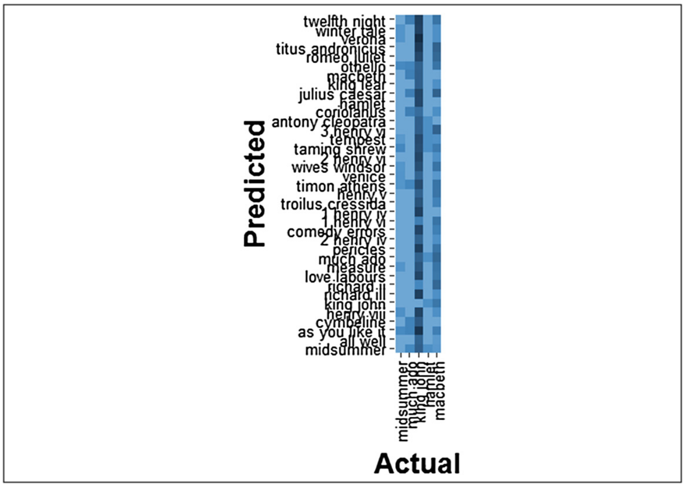


Fig. 8. Visualization of our fixed dialogue search results provided in the form of a depleted confusion matrix $\in \mathbb{R}^{37\times5}$.

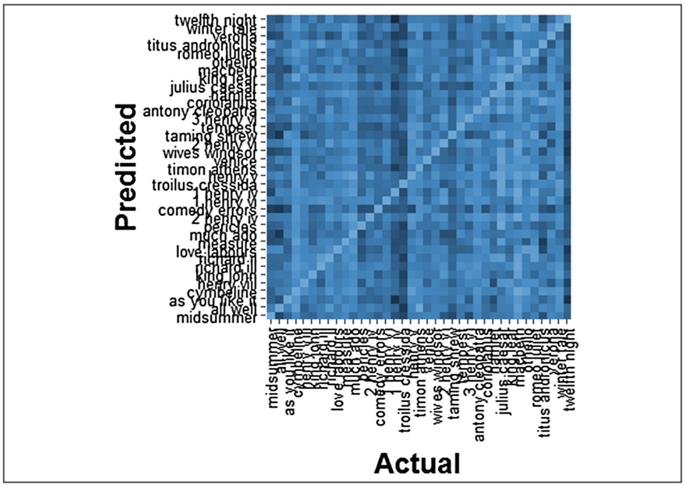


Fig. 9. Visualization of our randomly reordered sub-dialogue search results provided in the form of a fully populated confusion-matrix $\in \mathbb{R}^{37\times37}$.

plays. Distributed non uniformly, our token based keyphrases are evidently biased towards affiliation with plays of the largest content. Figure 10 shows non-zero query allocations that occupy 36 out of 37 plays, excluding the Comedy of Errors title. As a preprocess step, we iterated over the extended search matrix of dimensionality (37×200) and identified the play that is closest to a given query. We followed by averaging the distances in the case of multiple queries per play, and ended up with a reduced search matrix $\in \mathbb{R}^{37\times36}$. Figure 11 shows the

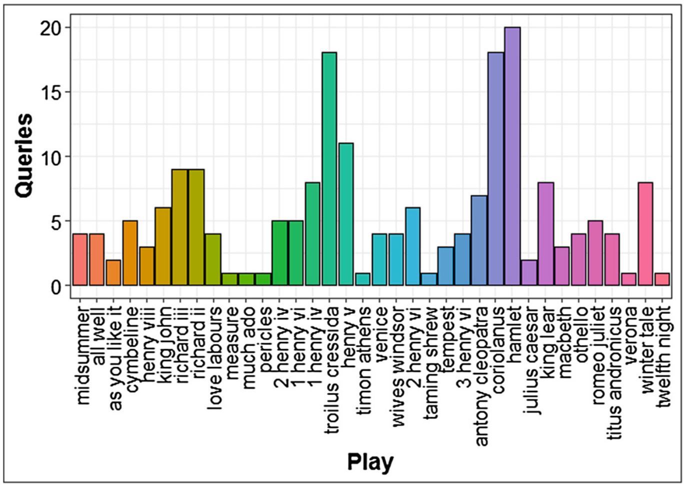


Fig. 10. Distribution of queries in random token search, presented across 36 Shakespeare titles. The larger the playscript content, the higher the query count.

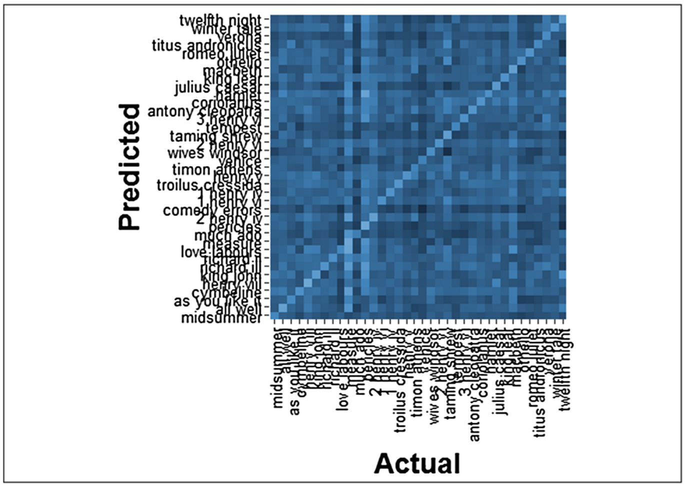


Fig. 11. Visualization of our random token search results provided in the form of a reduced dimensionality confusion-matrix $\in \mathbb{R}^{37 \times 36}$.

results of random token search as the straddling bright line along the diagonal of the confusion matrix highlights our best ranks. In this experiment, search scores are expected to be diverse and span a rather wide range of $[0.23, 0.87]$.

In Table 5, we list computational runtime of our implementation for key tasks in performing linguistic clustering and search. All reported figures are in seconds and were obtained by running our software single-threaded on a Windows 10 mobile device, with Intel 4th generation Core$^{\text{TM}}$ processor at 1.8 GHz and 8 GB

Table 5. Running time for key computational tasks in performing clustering and search over the Shakespeare corpus. All figures are shown in seconds.

Task	Min	Max	Mean	Total
Hierarchy Formation	0.58	1.44	0.91	33.53
Distance Matrix	0.83	80.08	56.19	2,079.35
Keyphrase Query	13.48	18.66	16.65	616.05

of memory. Play hierarchy construction is linear with the number of dialogues per play, and given a fairly balanced distribution over playscripts (Fig. 6), the play of Othello claimed the slowest to generate the data structure at 1.44 s. The distance matrix item shows the time to compute a set of similarities for one play paired with each of the rest of the plays in the Shakespeare collection. On average, play-to-play distance derivation amortized across $|T| = 37$ plays takes about 1.51 s. Launching a keyphrase query task typically involves a dialogue-to-play similarity operator that is linear in the total number of dialogues for the entire play collection. Query response times are shown for each search episode and are uniformly impartial to the keyphrase originating play, as evidenced by a small standard deviation of two percent of the mean. The total column of Table 5 further accumulates individual play processing times and is roughly the mean column value multiplied by the number of plays, $|T| = 37$.

6 Conclusions

In this study, we have demonstrated the apparent potential in a hierarchical representation of word embeddings to conduct effective play level clustering and search. We trained our system on a 37-dramatic-play corpus that comprises a vocabulary of about 2 million tokens, and generated our own word vectors for each of our experimental choices of model-hyperparameter configurations. We showed that the CBOW neural model outperforms skip-gram for the linguistic tasks we performed, and furthermore, clustering under CBOW proved sustainable to modifying the context window size in a fairly large extent. To evaluate any-pair semantic similarity of both play-to-play and query-to-play, we used a simple and generic distance metric [5] between a pair of word vector sets, each of up to tens of thousands elements, that disambiguates non-matching matrix dimensionality. We reported robust empirical results on our tasks for deploying state-of-the-art unsupervised learning of word representations.

At first observation, our hierarchical representation of plays might appear greedy storage wise, and rather than a matrix interpretation, we could have resorted to a more compact format by averaging all the dialogue word vectors and produce a single dialogue vector. While this approach seems plausible for the clustering tasks to both reduce footprint and streamline computations, the data loss incurred by doing so adversely impacted the performance of our search tasks. To address this shortcoming, our experimental choice of a modest 10-dimensional word vector appears as a reasonable system-design trade-off that aids to circumvent excessive usage of memory space. On average, there are 17,509 word vectors per-play (Fig. 6) and hence storage space for the entire Shakespeare play-suite is at a moderate $37 \times 17,509 \times 10 \times 4 \approx 26$ MB that is proportional to 34 MB area claimed by the bible book set [5].

To the best of our knowledge and based on literature published to date, we are unaware of semantic analysis systems with similar goals to evenhandedly contrast our results against. The more recent work by Yang *et al.* [27], proposed a hierarchical attention network for document classification. Their neural model explores attention mechanisms at both a word and sentence levels in an attempt to differentiate content importance when constructing a document vector representation. However, for evaluation their work focuses primarily on topic classification of short user-review snippets. Unlike our system that reasons semantic relatedness between any full-length plays. On the other hand, Jiang *et al.* [13] skip the sentence level construct altogether and combine a set of word vectors to directly represent a complete Yelp review. In their report, there is limited exposure to fine-grained control over the underlying neural models to show performance impact on business clustering.

Given that the training of word vectors is a one time process, a natural progression of our work is to optimize the core computations of constructing the distance matrix and performing a keyphrase query. The inherent independence of deriving similarity matrix elements and separating play search rankings lets us leverage parallel execution, and we expect to reduce our runtime complexity markedly. For a larger number of corpus plays, to uphold efficient hierarchy access our system would benefit from word vector caching, and we contend that projecting the distance matrix onto a three-dimensional embedding space is essential to improve cluster perception for analysis. Lastly, we seek to apply our play distance matrix directly to methods that partition objects around medoids [14] and potentially avoid outliers.

Acknowledgements. We would like to thank the anonymous reviewers for their insightful suggestions and feedback.

References

1. Agirre, E., Alfonseca, E., Hall, K., Kravalova, J., Paca, M., Soroa, A.: A study on similarity and relatedness using distributional and WordNet-based approaches. In: Human Language Technologies: North American Chapter of the Association for Computational Linguistics (NAACL), Stroudsburg, PA, pp. 19–27 (2009)
2. Baeza-Yates, R., Ribeiro-Neto, B. (eds.): Modern Information Retrieval. ACM Press Series/Addison Wesley, Essex (1999)
3. Baroni, M., Dinu, G., Kruszewski, G.: Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. In: Annual Meeting of the Association for Computational Linguistics (ACL), pp. 238–247 (2014)
4. Bengio, Y., Ducharme, R., Vincent, P., Janvin, C.: A neural probabilistic language model. Mach. Learn. Res. (JMLR) **3**, 1137–1155 (2003)

5. Bleiweiss, A.: A hierarchical book representation of word embeddings for effective semantic clustering and search. In: Agents and Artificial Intelligence (ICAART), pp. 154–163. Porto, Portugal (2017)
6. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Mach. Learn. Res. (JMLR) **12**, 2493–2537 (2011)
7. Cormen, T.H., Leiserson, C.H., Rivest, R.L., Stein, C.: Introduction to Algorithms. MIT Press/McGraw-Hill Book Company, Cambridge (1990)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Unsupervised learning and clustering. In: Pattern Classification, pp. 517–601. Wiley, New York (2001)
9. Fu, R., Guo, J., Qin, B., Che, W., Wang, H., Liu, T.: Learning semantic hierarchies via word embeddings. In: Annual Meeting of the Association for Computational Linguistics (ACL), Baltimore, MD, pp. 1199–1209 (2014)
10. Guo, J., Che, W., Wang, H., Liu, T.: Revisiting embedding features for simple semi-supervised learning. In: Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 110–120 (2014)
11. Hofmann, T., Buhmann, J.: Multidimensional scaling and data clustering. In: Advances in Neural Information Processing Systems, pp. 459–466. MIT Press, Cambridge (1995)
12. Iyyer, M., Boyd-Graber, J., Claudino, L., Socher, R., Daume, H.: A neural network for factoid question answering over paragraphs. In: Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 633–644 (2014)
13. Jiang, R., Liu, Y., Xu, K.: A General Framework for Text Semantic Analysis and Clustering on Yelp Reviews (2015). http://cs229.stanford.edu/proj2015/003_report.pdf
14. Kaufman, L., Rousseeuw, P.J. (eds.): Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (1990)
15. Kim, Y.: Convolutional neural networks for sentence classification. In: Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1746–1751 (2014)
16. Le, Q.V., Mikolov, T.: Distributed representations of sentences and documents. CoRR abs/1405.4053 (2014). https://arxiv.org/abs/1405.4053
17. Manning, C.D., Raghavan, P., Schutze, H.: Introduction to Information Retrieval. Cambridge University Press, Cambridge (2008)
18. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013a). http://arxiv.org/abs/1301.3781
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119. Curran Associates Inc, Red Hook (2013b)
20. Pennington, J., Socher, R., Manning, C.D.: GloVe: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, pp. 1532–1543 (2014)
21. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2013). http://www.R-project.org/
22. Salton, G., Wong, A., Yang, C.S.: A vector space model for automatic indexing. Commun. ACM **18**(11), 613–620 (1975)
23. Shakespeare, W.: Project Gutenberg: The complete works of William Shakespeare (1994). http://www.gutenberg.org/ebooks/100

24. Socher, R., Perelygin, A., Wu, J.Y., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C.: Recursive deep models for semantic compositionality over a sentiment treebank. In: Empirical Methods in Natural Language Processing (EMNLP), Seattle, WA, pp. 1631–1642 (2013)
25. Torgerson, W.S.: Theory and Methods of Scaling. Wiley, New York (1958)
26. Turney, P.D., Pantel, T.: From frequency to meaning: vector space models of semantics. Artif. Intell. Res. (JAIR) **37**, 141–188 (2010)
27. Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E.: Hierarchical attention networks for document classification. In: Human Language Technologies: North American Chapter of the Association for Computational Linguistics, pp. 1480–1489 (2016)