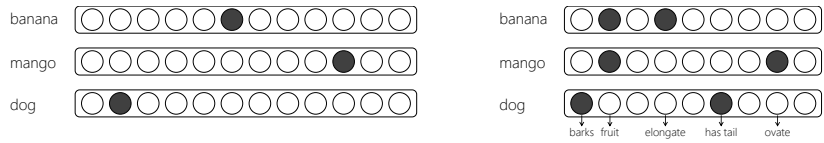
3

Unsupervised learning of term representations

3.1 A tale of two representations

Vector representations are fundamental to both information retrieval and machine learning. In IR, terms are typically the smallest unit of representation for indexing and retrieval. Therefore, many IR models—both non-neural and neural—focus on learning good vector representations of terms. Different vector representations exhibit different levels of generalization—some consider every term as a distinct entity while others learn to identify common attributes. Different representation schemes derive different notions of similarity between terms from the definition of the corresponding vector spaces. Some representations operate over fixed-size vocabularies, while the design of others obviate such constraints. They also differ on the properties of compositionality that defines how representations for larger units of information, such as passages and documents, can be derived from individual term vectors. These are some of the important considerations for choosing a term representation suitable for a specific task.

Local representations Under local (or *one-hot*) representations, every term in a fixed size vocabulary T is represented by a binary vector $\vec{v} \in \{0, 1\}^{|T|}$, where only one of the values in the vector is one and all the others are set to zero. Each position in the vector $\vec{v}$ corresponds to a term. The term "banana", under this representation, is given by a vector that has the value one in the position corresponding to "banana" and zero everywhere else. Similarly, the



(a) Local representation **(b)** Distributed representation

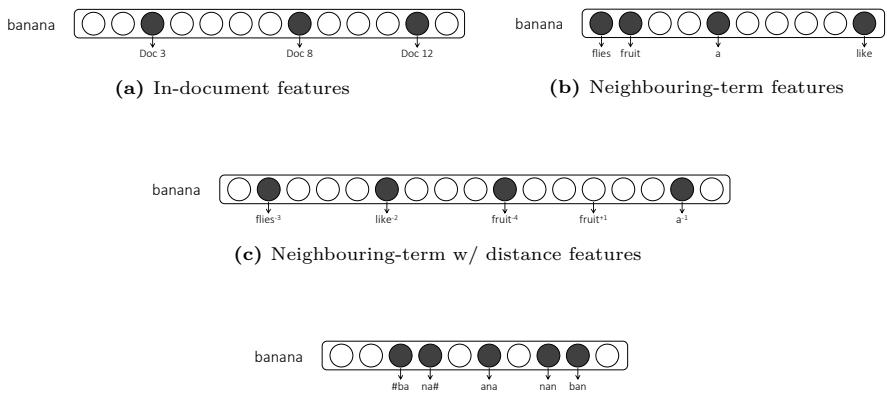
Figure 3.1: Under local representations the terms "banana", "mango", and "dog" are distinct items. But distributed vector representations may recognize that "banana" and "mango" are both fruits, but "dog" is different.

terms "mango" and "dog" are represented by setting different positions in the vector to one. Figure 3.1a highlights that under this scheme each term is a unique entity, and "banana" is as distinct from "dog" as it is from "mango". Terms outside of the vocabulary either have no representation or are denoted by a special "UNK" symbol under this scheme.

Distributed representations Under distributed representations every term is represented by a vector $\vec{v} \in \mathbb{R}^{|k|}$. $\vec{v}$ can be a sparse or a dense vector—a vector of hand-crafted features or a latent representation in which the individual dimensions are not interpretable in isolation. The key underlying hypothesis for any distributed representation scheme, however, is that by representing a term by its attributes allows for defining some notion of similarity between the different terms based on the chosen properties. For example, in Figure 3.1b "banana" is more similar to "mango" than "dog" because they are both fruits, but yet different because of other properties that are not shared between the two, such as shape.

Under a local or one-hot representation every item is distinct. But when items have distributed or feature based representation, then the similarity between two items is determined based on the similarity between their features.

A key consideration in any feature based distributed representation is the choice of the features themselves. One approach involves representing terms by features that capture their distributional properties. This is motivated by



(a) In-document features

(b) Neighbouring-term features

(c) Neighbouring-term w/ distance features

(d) Character-trigraph features

Figure 3.2: Examples of different feature-based distributed representations of the term "banana". The representations in (a), (b), and (c) are based on external contexts in which the term frequently occurs, while (d) is based on properties intrinsic to the term. The representation scheme in (a) depends on the documents containing the term while the scheme shown in (b) and (c) depends on other terms that appears in its neighbourhood. The scheme (b) ignores inter-term distances. Therefore, in the sentence "Time flies like an arrow; fruit flies like a banana", the feature "fruit" describes both the terms "banana" and "arrow". However, in the representation scheme of (c) the feature "fruit^{-4}" is positive for "banana", and the feature "fruit^{+1}" for "arrow".

the *distributional hypothesis* (Harris, 1954) that states that terms that are used (or occur) in similar context tend to be semantically similar. Firth (1957) famously purported this idea of *distributional semantics*[1] by stating "*a word is characterized by the company it keeps*". However, the distribution of different types of context may model different semantics of a term. Figure 3.2 shows three different sparse vector representations of the term "banana" corresponding to different distributional feature spaces—documents containing the term (*e.g.*, LSA (Deerwester *et al.*, 1990)), neighbouring terms in a window (*e.g.*, HAL (Lund and Burgess, 1996), COALS (Rohde *et al.*, 2006), and (Bullinaria and Levy, 2007)), and neighbouring terms with distance (*e.g.*, (Levy *et al.*, 2014)). Finally, Figure 3.2d shows a vector representation of "banana" based on the

[1]Readers should take note that while many distributed representations take advantage of *distributional* properties, the two concepts are not synonymous. A term can have a distributed representation based on non-distributional features—*e.g.*, parts of speech classification and character trigraphs in the term.

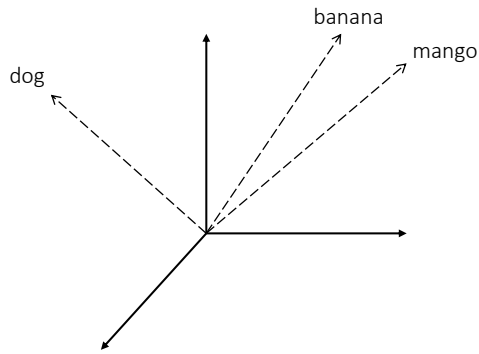


Figure 3.3: A vector space representation of terms puts "banana" closer to "mango" because they share more common attributes than "banana" and "dog".

character trigraphs in the term itself—instead of external contexts in which the term occurs. In §3.2 we will discuss how choosing different distributional features for term representation leads to different nuanced notions of semantic similarity between them. When the vectors are high-dimensional, sparse, and based on observable features we refer to them as *observed* (or *explicit*) vector representations (Levy *et al.*, 2014). When the vectors are dense, small ($k \ll |T|$), and learnt from data then we instead refer to them as *latent* vector spaces, or *embeddings*. In both observed and latent vector spaces, several distance metrics can be used to define the similarity between terms, although cosine similarity is commonly used.

$$sim(\vec{v}_i, \vec{v}_j) = cos(\vec{v}_i, \vec{v}_j) = \frac{\vec{v}_i^{\intercal} \vec{v}_j}{\|\vec{v}_i\| \|\vec{v}_j\|} \tag{3.1}$$

Most embeddings are learnt from observed features, and hence the discussions in §3.2 about different notions of similarity are also relevant to the embedding models. In §3.3 and §3.4 we discuss observed and latent space representations. In the context of neural models, distributed representations generally refer to learnt embeddings. The idea of 'local' and 'distributed' representations has a specific significance in the context of neural networks. Each concept, entity, or term can be represented within a neural network by the activation of a single neuron (local representation) or by the combined pattern of activations of several neurons (distributed representation) (Hinton, 1984).

Finally, with respect to *compositionality*, it is important to understand that distributed representations of items are often derived from local or distributed representation of its parts. For example, a document can be represented by the sum of the one-hot vectors or embeddings corresponding to the terms in the

Table 3.1: A toy corpus of short documents that we consider for the discussion on different notions of similarity between terms under different distributed representations. The choice of the feature space that is used for generating the distributed representation determines which terms are closer in the vector space, as shown in Figure 3.4.

Sample documents			
doc 01	Seattle map	doc 09	Denver map
doc 02	Seattle weather	doc 10	Denver weather
doc 03	Seahawks jerseys	doc 11	Broncos jerseys
doc 04	Seahawks highlights	doc 12	Broncos highlights
doc 05	Seattle Seahawks Wilson	doc 13	Denver Broncos Lynch
doc 06	Seattle Seahawks Sherman	doc 14	Denver Broncos Sanchez
doc 07	Seattle Seahawks Browner	doc 15	Denver Broncos Miller
doc 08	Seattle Seahawks Ifedi	doc 16	Denver Broncos Marshall

document. The resultant vector, in both cases, corresponds to a distributed bag-of-terms representation. Similarly, the character trigraph representation of terms in Figure 3.2d is simply an aggregation over the one-hot representations of the constituent trigraphs.

3.2 Notions of similarity

Any vector representation inherently defines some notion of relatedness between terms. Is "Seattle" closer to "Sydney" or to "Seahawks"? The answer depends on the type of relationship we are interested in. If we want terms of similar *type* to be closer, then "Sydney" is more similar to "Seattle" because they are both cities. However, if we are interested to find terms that co-occur in the same document or passage, then "Seahawks"—Seattle's football team—should be closer. The former represents a *typical*, or type-based notion of similarity while the latter exhibits a more *topical* sense of relatedness.

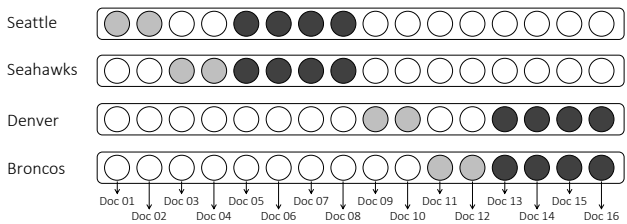
If we want to compare "Seattle" with "Sydney" and "Seahawks based on their respective vector representations, then the underlying feature space needs to align with the notion of similarity that we are interested in. It is, therefore, important for the readers to build an intuition about the choice of features and the notion of similarity they encompass. This can be demonstrated by using a toy corpus, such as the one in Table 3.1. Figure 3.4a shows that the "in documents" features naturally lend to a topical sense of similarity between the terms, while the "neighbouring terms with distances" features in Figure 3.4c gives rise to a more typical notion of relatedness. Using "neighbouring terms" without the inter-term distances as features, however, produces a

mixture of topical and typical relationships. This is because when the term distances (denoted as superscripts) are considered in the feature definition then the document "Seattle Seahawks Wilson" produces the bag-of-features $\{Seahawks^{+1}, Wilson^{+2}\}$ for "Seattle" which is non-overlapping with the bag-of-features $\{Seattle^{-1}, Wilson^{+1}\}$ for "Seahawks". However, when the feature definition ignores the term-distances then there is a partial overlap between the bag-of-features $\{Seahawks, Wilson\}$ and $\{Seattle, Wilson\}$ corresponding to "Seattle" and "Seahawks", respectively. The overlap increases when a larger window-size over the neighbouring terms is employed pushing the notion of similarity closer to a topical definition. This effect of the windows size on the latent vector space was reported by Levy and Goldberg (2014) in the context of term embeddings.

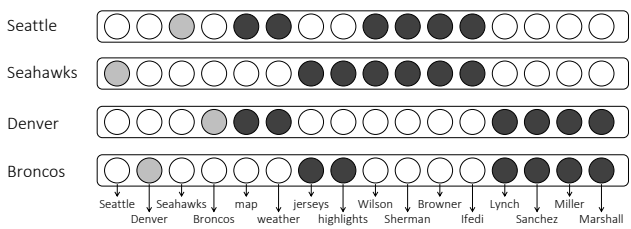
Different vector representations capture different notions of similarity between terms. "Seattle" may be closer to either "Sydney" (*typically* similar) or "Seahawks" (*topically* similar) depending on the choice of vector dimensions.

Readers should note that the set of all inter-term relationships goes beyond the two notions of typical and topical that we discuss in this section. For example, vector representations could cluster terms closer based on linguistic styles—*e.g.*, terms that appear in thriller novels versus in children's rhymes, or in British versus American English. However, the notions of typical and topical similarities frequently come up in discussions in the context of many IR and NLP tasks—sometimes under different names such as *Paradigmatic* and *Syntagmatic* relations[2]—and the idea itself goes back at least as far as Saussure (De Saussure, 1916; Harris, 2001; Chandler, 1994; Sahlgren, 2006).

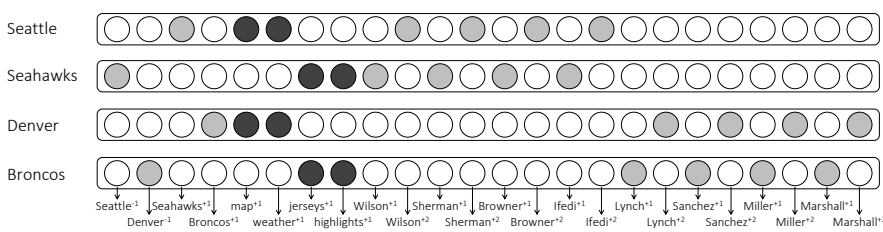
[2]Interestingly, the notion of Paradigmatic (typical) and Syntagmatic (topical) relationships show up almost universally—not just in text. In vision, for example, the different images of "nose" are typically similar to each other, while sharing topical relationship with images of "eyes" and "ears". Curiously, Barthes (1977) extended the analogy to garments. Paradigmatic relationships exist between items of the same type (*e.g.*, different style of boots) and the proper Syntagmatic juxtaposition of items from these different Paradigms—from hats to boots—forms a fashionable ensemble.



(a) "In-documents" features



(b) "Neighbouring terms" features



(c) "Neighbouring terms w/ distances" features

Figure 3.4: The figure shows different distributed representations for the four terms—"Seattle", "Seahawks", "Denver", and "Broncos"—based on the toy corpus in Table 3.1. Shaded circles indicate non-zero values in the vectors—the darker shade highlights the vector dimensions where more than one vector has a non-zero value. When the representation is based on the documents that the terms occur in then "Seattle" is more similar to "Seahawks" than to "Denver". The representation scheme in (a) is, therefore, more aligned with a topical notion of similarity. In contrast, in (c) each term is represented by a vector of neighbouring terms—where the distances between the terms are taken into consideration—which puts "Seattle" closer to "Denver" demonstrating a typical, or type-based, similarity. When the inter-term distances are ignored, as in (b), a mix of typical and topical similarities is observed. Finally, it is worth noting that neighbouring-terms based vector representations leads to similarities between terms that do not necessarily occur in the same document, and hence the term-term relationships are less sparse than when only in-document features are considered.

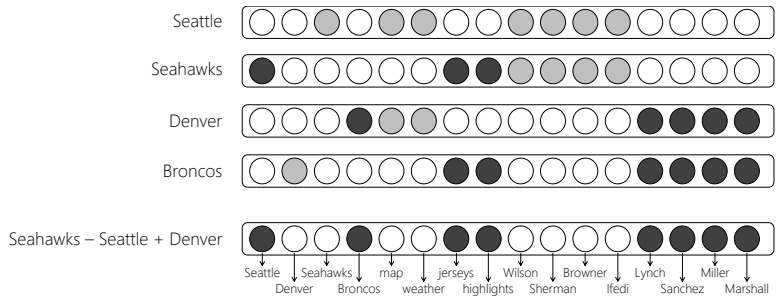


Figure 3.5: A visual demonstration of term analogies via simple vector algebra. The shaded circles denote non-zero values. Darker shade is used to highlight the non-zero values along the vector dimensions for which the output of $\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver}$ is positive. The output vector is closest to $\vec{v}_{Broncos}$ as shown in this toy example.

3.3 Observed feature spaces

Observed feature space representations can be broadly categorized based on their choice of distributional features (*e.g.*, in documents, neighbouring terms with or without distances, *etc.*) and different weighting schemes (*e.g.*, TF-IDF, positive pointwise mutual information, *etc.*) applied over the raw counts. We direct the readers to (Turney and Pantel, 2010; Baroni and Lenci, 2010) which are good surveys of many existing observed vector representation schemes.

Levy *et al.* (2014) demonstrated that explicit vector representations are amenable to the term analogy task using simple vector operations. A term analogy task involves answering questions of the form "*man* is to *woman* as *king* is to _____?"—the correct answer to which in this case happens to be "queen". In NLP, term analogies are typically performed by simple vector operations of the following form followed by a nearest-neighbour search,

$$\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver} \approx \vec{v}_{Broncos} \tag{3.2}$$

It may be surprising to some readers that the vector obtained by the simple algebraic operations $\vec{v}_{Seahawks} - \vec{v}_{Seattle} + \vec{v}_{Denver}$ produces a vector close to the vector $\vec{v}_{Broncos}$. We present a visual intuition of why this works in practice in Figure 3.5, but we refer the readers to (Levy *et al.*, 2014; Arora *et al.*, 2015) for a more rigorous mathematical handling of this subject.

3.4 Latent feature spaces

While observed vector spaces based on distributional features can capture interesting relationships between terms, they have one big drawback—the resultant representations are highly sparse and high-dimensional. The number of dimensions, for example, may be the same as the vocabulary size, which is unwieldy for most practical tasks. An alternative is to learn lower dimensional representations that retains useful attributes from the observed feature spaces.

An *embedding* is a representation of items in a new space such that the properties of—and the relationships between—the items are preserved. Goodfellow *et al.* (2016) articulate that the goal of an embedding is to generate a *simpler* representation—where simplification may mean a reduction in the number of dimensions, an increase in the sparseness of the representation, disentangling the principle components of the vector space, or a combination of these goals. In the context of term embeddings, the explicit feature vectors—like those discussed in §3.3—constitutes the original representation. An embedding trained from these features assimilate the properties of the terms and the inter-term relationships observable in the original feature space.

An embedding is a representation of items in a new space such that the properties of—and the relationships between—the items are preserved from the original representation.

Common approaches for learning embeddings include either factorizing the term-feature matrix (*e.g.*LSA (Deerwester *et al.*, 1990)) or using gradient descent based methods that try to predict the features given the term (*e.g.*, (Bengio *et al.*, 2003; Mikolov *et al.*, 2013a)). Baroni *et al.* (2014) empirically demonstrate that these feature-predicting models that learn lower dimensional representations, in fact, also perform better than explicit counting based models on different tasks—possibly due to better generalization across terms—although some counter evidence the claim of better performances from embedding models have also been reported in the literature (Levy *et al.*, 2015).

The sparse feature spaces of §3.3 are easier to visualize and leads to more intuitive explanations—while their latent counterparts may be more practically useful. Therefore, it may be useful to *think sparse, but act dense* in many scenarios. In the rest of this section, we will describe some of these neural and non-neural latent space models.

Latent Semantic Analysis (LSA) LSA (Deerwester *et al.*, 1990) involves performing *singular value decomposition* (SVD) (Golub and Reinsch, 1970) on a term-document (or term-passage) matrix X to obtain its low-rank approximation (Markovsky, 2011). SVD on X involves solving $X = U\Sigma V^T$, where U and V are orthogonal matrices and Σ is a diagonal matrix.[3]

$$\begin{matrix} & \overset{X}{(\vec{d}_j)} & & \overset{U}{} & & \overset{\Sigma}{} & & \overset{V^\intercal}{(\vec{d}_j)} \\ & \downarrow & & & & & & \downarrow \\ (\vec{t}_i^{\,\intercal})\rightarrow & \begin{bmatrix} x_{1,1} & \dots & x_{1,|D|} \\ \vdots & \ddots & \vdots \\ x_{|T|,1} & \dots & x_{|T|,|D|} \end{bmatrix} & = (\vec{t}_i^{\,\intercal})\rightarrow & \begin{bmatrix} \begin{bmatrix} \\ \vec{u}_1 \\ \\ \end{bmatrix} & \dots & \begin{bmatrix} \\ \vec{u}_l \\ \\ \end{bmatrix} \end{bmatrix} & \cdot & \begin{bmatrix} \sigma_1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma_l \end{bmatrix} & \cdot & \begin{bmatrix} \begin{bmatrix} \vec{v}_1 \end{bmatrix} \\ \vdots \\ \begin{bmatrix} \vec{v}_l \end{bmatrix} \end{bmatrix} \end{matrix} \tag{3.3}$$

$\sigma_1, \dots, \sigma_l$, $\vec{u}_1, \dots, \vec{u}_l$, and $\vec{v}_1, \dots, \vec{v}_l$ are the singular values, and the left and the right singular vectors, respectively. The k largest singular values—and corresponding singular vectors from U and V—is the rank k approximation of X ($X_k = U_k \Sigma_k V_k^T$) and $\Sigma_k \vec{t}_i$ is the embedding for the i^{th} term.

While LSA operate on a term-document matrix, matrix factorization based approaches can also be applied to term-term matrices (Rohde *et al.*, 2006; Bullinaria and Levy, 2012; Lebret and Collobert, 2013).

Probabilistic Latent Semantic Analysis (PLSA) PLSA (Hofmann, 1999) learns low-dimensional representations of terms and documents by modelling their co-occurrence $p(t, d)$ as follows,

$$p(t, d) = p(d) \sum_{c \in C} p(c|d) P(t|c) \tag{3.4}$$

where, C is the set of latent topics—and the number of topics $|C|$ is a hyperparameter of the model. Both $p(c|d)$ and $P(t|c)$ are modelled as multinomial distributions and their parameters are typically learned using the EM algorithm (Dempster *et al.*, 1977). After learning the parameters of the model, a term t_i can be represented as a distribution over the latent topics $[p(c_0|t_i), \dots, p(c_{|C|-1}|t_i)]$. In a related approach called Latent Dirichlet Allocation (LDA) (Blei *et al.*, 2003), each document is represented by a Dirichlet prior instead of a fixed variable.

Neural term embedding models are typically trained by setting up a prediction task. Instead of factorizing the term-feature matrix—as in LSA—neural

[3]The matrix visualization is taken from https://en.wikipedia.org/wiki/Latent_semantic_analysis.

models are trained to predict the term from its features. The model learns dense low-dimensional representations in the process of minimizing the prediction error. These approaches are based on the *information bottleneck method* (Tishby *et al.*, 2000)—discussed more in §6.2—with the low-dimensional representations acting as the bottleneck. The training data may contain many instances of the same term-feature pair proportional to their frequency in the corpus (*e.g.*, word2vec (Mikolov *et al.*, 2013a)), or their counts can be pre-aggregated (*e.g.*, GloVe (Pennington *et al.*, 2014)).

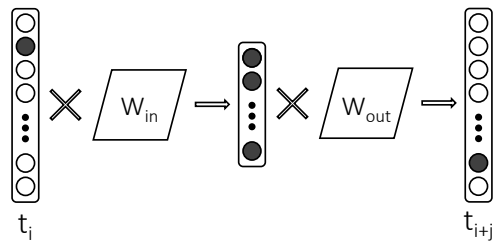
Instead of factorizing the term-feature matrix, neural models learn embeddings by setting up a feature prediction task and employ architectures motivated by the *information bottleneck* principle.

Word2vec For word2vec (Mikolov *et al.*, 2013a; Mikolov *et al.*, 2013b; Mikolov *et al.*, 2013c; Goldberg and Levy, 2014; Rong, 2014), the features for a term are made up of its neighbours within a fixed size window over the text. The *skip-gram* architecture (see Figure 3.6a) is a simple one hidden layer neural network. Both the input and the output of the model are one-hot vectors and the loss function is as follows,

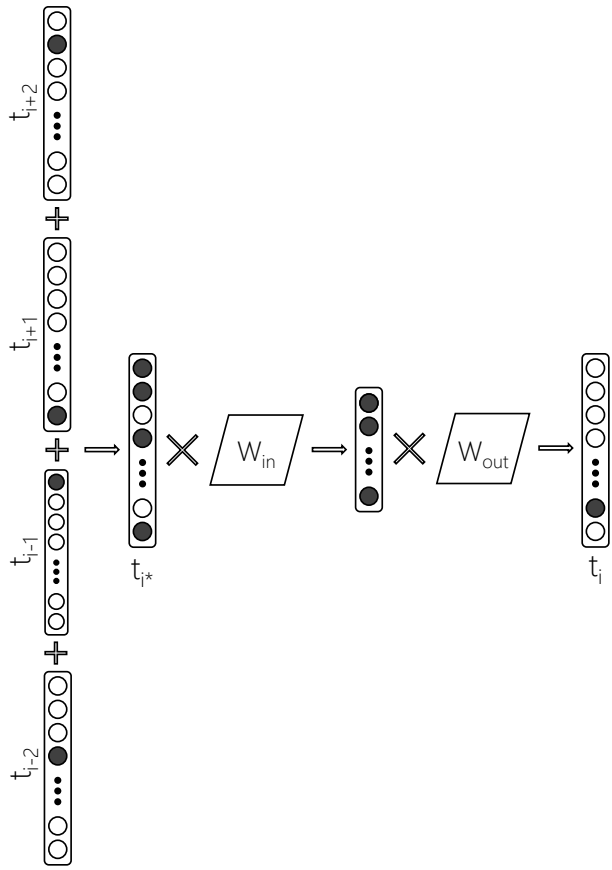
$$\mathcal{L}_{skip-gram} = -\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-c\leq j\leq +c, j\neq 0} log(p(t_{i+j}|t_i)) \tag{3.5}$$

$$\text{where,} \qquad p(t_{i+j}|t_i) = \frac{\exp\left((W_{out}\vec{v}_{t_{i+j}})^\intercal (W_{in}\vec{v}_{t_i})\right)}{\sum_{k=1}^{|T|}\exp\left((W_{out}\vec{v}_{t_k})^\intercal (W_{in}\vec{v}_{t_i})\right)} \tag{3.6}$$

S is the set of all windows over the training text and c is the number of neighbours we want to predict on either side of the term t_i. The denominator for the softmax function for computing $p(t_{i+j}|t_i)$ sums over all the terms in the vocabulary. This is prohibitively costly and in practice either hierarchical-softmax (Morin and Bengio, 2005) or negative sampling is employed, which we discuss more in §5.2. Note that the model has two different weight matrices W_{in} and W_{out} that constitute the learnable parameters of the models. W_{in} gives us the IN embeddings corresponding to the input terms and W_{out} corresponds to the OUT embeddings for the output terms. Generally, only W_{in} is used and W_{out} is discarded after training. We discuss an IR application that makes use of both the IN and the OUT embeddings in §4.1.



(a) Skip-gram



(b) Continuous bag-of-words (CBOW)

Figure 3.6: The (a) skip-gram and the (b) continuous bag-of-words (CBOW) architectures of word2vec. The architecture is a neural network with a single hidden layer whose size is much smaller than that of the input and the output layers. Both models use one-hot representations of terms in the input and the output. The learnable parameters of the model comprise of the two weight matrices W_{in} and W_{out} that corresponds to the embeddings the model learns for the input and the output terms, respectively. The skip-gram model trains by minimizing the error in predicting a term given one of its neighbours. The CBOW model, in contrast, predicts a term from a bag of its neighbouring terms.

The *continuous bag-of-words* (CBOW) architecture (see Figure 3.6b) is similar to the skip-gram model, except that the task is to predict the middle term given all the neighbouring terms in the window. The CBOW model creates a single training sample with the sum of the one-hot vectors of the neighbouring terms as input and the one-hot vector $\vec{v}_{t_i}$—corresponding to the middle term—as the expected output. Contrast this with the skip-gram model that creates $2 \times c$ samples by individually pairing each neighbouring term with the middle term. During training, the skip-gram model trains slower than the CBOW model (Mikolov *et al.*, 2013a) because it creates more training samples from the same windows of text.

$$\mathcal{L}_{CBOW} = -\frac{1}{|S|}\sum_{i=1}^{|S|} log(p(t_i|t_{i-c},\ldots,t_{i-1},t_{i+1},\ldots,t_{i+c})) \tag{3.7}$$

Word2vec gained particular popularity for its ability to perform term analogies using simple vector algebra, similar to what we discussed in §3.3. For domains where the interpretability of the embeddings is important, Sun *et al.* (2016b) introduced an additional constraint in the loss function to encourage more sparseness in the learnt representations.

$$\mathcal{L}_{sparse-CBOW} = \mathcal{L}_{sparse-CBOW} - \lambda \sum_{t \in T} \|\vec{v}_t\|_1 \tag{3.8}$$

GloVe The skip-gram model trains on individual term-neighbour pairs. If we aggregate all the training samples such that x_{ij} is the frequency of the pair $\langle t_i, t_j \rangle$ in the training data, then the loss function changes to,

$$\mathcal{L}_{skip-gram} = -\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} x_{ij} log(p(t_j|t_i)) \tag{3.9}$$

$$= -\sum_{i=1}^{|T|} x_i \sum_{j=1}^{|T|} \frac{x_{ij}}{x_i} log(p(t_j|t_i)) \tag{3.10}$$

$$= -\sum_{i=1}^{|T|} x_i \sum_{j=1}^{|T|} \bar{p}(t_j|t_i) log(p(t_j|t_i)) \tag{3.11}$$

$$= \sum_{i=1}^{|T|} x_i H(\bar{p}(t_j|t_i), p(t_j|t_i)) \tag{3.12}$$

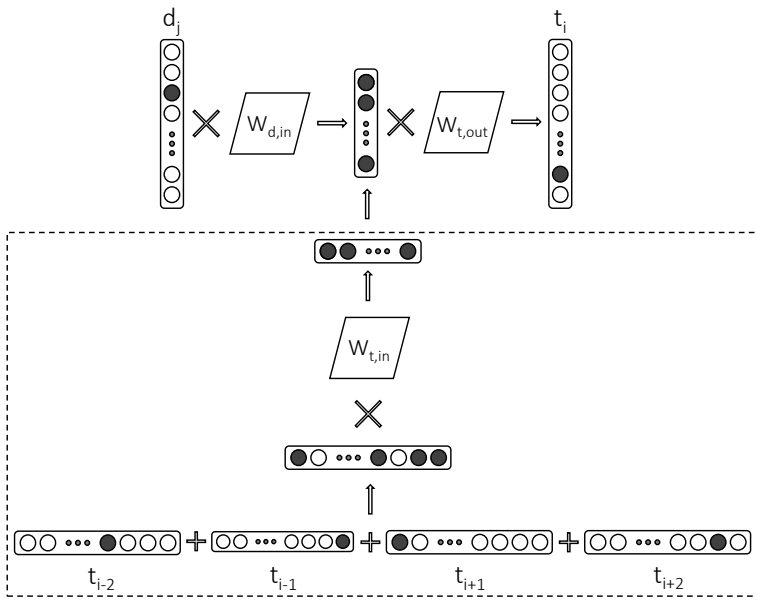


Figure 3.7: The paragraph2vec architecture as proposed by Le and Mikolov (2014) trains by predicting a term given a document (or passage) ID containing the term. By trying to minimize the prediction error, the model learns an embedding for the term as well as for the document. In some variants of the architecture, optionally the neighbouring terms are also provided as input—as shown in the dotted box.

$H(\dots)$ is the cross-entropy error between the actual co-occurrence probability $\bar{p}(t_j|t_i)$ and the one predicted by the model $p(t_j|t_i)$. This is similar to the loss function for GloVe (Pennington *et al.*, 2014) if we replace the cross-entropy error with a squared-error and apply a saturation function $f(\dots)$ over the actual co-occurrence frequencies.

$$\mathcal{L}_{GloVe} = -\sum_{i=1}^{|T|}\sum_{j=1}^{|T|} f(x_{ij})(log(x_{ij} - \vec{v}_{w_i}^{\intercal}\vec{v}_{w_j}))^2 \tag{3.13}$$

$$\tag{3.14}$$

where,

$$f(x) = \begin{cases} (x/x_{max})^{\alpha}, & \text{if} x \leq x_{max} \\ 1, & \text{otherwise} \end{cases} \tag{3.15}$$

GloVe is trained using AdaGrad (Duchi *et al.*, 2011). Similar to word2vec, GloVe also generates two different (IN and OUT) embeddings, but unlike word2vec it generally uses the sum of the IN and the OUT vectors as the embedding for each term in the vocabulary.

Paragraph2vec Following the popularity of word2vec (Mikolov *et al.*, 2013a; Mikolov *et al.*, 2013b), similar neural architectures (Le and Mikolov, 2014; Grbovic *et al.*, 2015b; Grbovic *et al.*, 2015a; Sun *et al.*, 2015; Ai *et al.*, 2016b; Ai *et al.*, 2016a) have been proposed that trains on term-document co-occurrences. The training typically involves predicting a term given the ID of a document or a passage that contains the term. In some variants, as shown in Figure 3.7, neighbouring terms are also provided as input. The key motivation for training on term-document pairs is to learn an embedding that is more aligned with a topical notion of term-term similarity—which is often more appropriate for IR tasks. The term-document relationship, however, tends to be more sparse (Yan *et al.*, 2013)—including neighbouring term features may compensate for some of that sparsity. In the context of IR tasks, Ai *et al.* (2016b) and Ai *et al.* (2016a) proposed a number of IR-motivated changes to the original Paragraph2vec (Le and Mikolov, 2014) model training—including, document frequency based negative sampling and document length based regularization.