Word and document representations

Two main approaches:

- Bag of Words (BoW) - traditional, older. Weak representation for words. Document well represented.
- Embedding approaches - relatively recent, neural n/w based. Rich representation for words. Document representation: work in progress.

Bag of words representation

- Bag of Words (BoW) model - simply records whether or not a word occurs in a document. It has an obvious and simple model for a document - namely the vocabulary sized vector which contains a 1 if the word occurs in the document else it is 0. These representations are usually very sparse.

Embedding based representation

- Embedding approaches embed a word in a low dimensional vector space using a simple neural network. The vector is learnt using large volumes of text and does not need supervision. The vector is usually much smaller than the vocabulary size. The representation is semantically richer.
- There are many proposals to embed documents in a vector space. Some of them use a neural network and learn a document vector much like a word vector is learnt. Other methods combine word vectors of words that occur in a document to form the document vector.
- Word vectors have become standard representations for words and are widely used. However, there is as yet no generally accepted representation for documents though several proposals exist.

Bag of Words (BoW) details

BoW is a class of representations that represent a document with a vocabulary sized vector. The representations differ in how word occurrence is encoded.

- V: vocabulary $\equiv$ the set of all words occurring in a set of documents or corpus. V is usually the set of words obtained after stop-word removal and stemming/lemmatization of words in the corpus.
- Each document is a binary vector $\mathbf{d}$ of size $|V|$. Each entry in $\mathbf{d}$ represents a word and is 1 if the word occurs in the document else it is 0. The word entries in $\mathbf{d}$ are in lexicographic order.
- This is the simplest kind of BoW representation - binary Bow or BBoW.

Example of BoW representation

$D1$: `The cat sat on the mat. The cat was white.`
$D2$: `The brown coloured dog was barking loudly at the cat.`
$D3$: `The mat was green in colour.`
$D4$: `The dog pulled the mat with his teeth. The cat still sat on the mat.`
$V \equiv \{$`bark, brown, cat, colour, dog, green, loud, mat, pull, sit, teeth, white`$\}$, $|V| = 12$.
$\mathbf{d_1} = (0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 1)$
$\mathbf{d_2} = (1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0)$
$\mathbf{d_3} = (0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0)$
$\mathbf{d_4} = (0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0)$

In practice the document vectors are sparse - many more 0s than 1s.

Term frequency

- Term frequency (**tf**) is the frequency of occurrence of a word in a document. This frequency is normalized by document size. The intuition is that the more frequently a word occurs in a document the more important it is. So, the value for it in the vector should be higher.

Many term frequency definitions exist:
Let $f_{t,d}$ be the number of times term t occurs in document d.

1. Raw tf: $tf(t,d) = f_{t,d}$.
2. Normalized tf: $tf(t,d) = \frac{f_{t,d}}{|d|}$.
3. Log scaled tf:

$$tf(t,d) = \begin{cases} 0 & f_{t,d} = 0 \\ ln(1 + f_{t,d}) & f_{t,d} > 0 \end{cases}$$

4. Augmented frequency tf: $tf(t,d) = 0.5 + 0.5\frac{f_{t,d}}{max\{f_{t',d}|t' \in d\}}$.

Earlier example with tf

$D1$: `The cat sat on the mat. The cat was white.`
$|D1| = 5$
$D2$: `The brown coloured dog was barking loudly at the cat.` $|D2| = 6$
$D3$: `The mat was green in colour.` $|D3| = 3$
$D4$: `The dog pulled the mat with his teeth. The cat still sat on the mat.` $|D4| = 7$
$V \equiv \{$`bark, brown, cat, colour, dog, green, loud, mat, pull, sit, teeth, white`$\}$, $|V| = 12$.

Normalized tf:
$\mathbf{d_1} = \frac{1}{5}(0, 0, 2, 0, 0, 0, 0, 1, 0, 1, 0, 1)$
$\mathbf{d_2} = \frac{1}{6}(1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0)$
$\mathbf{d_3} = \frac{1}{3}(0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0)$
$\mathbf{d_4} = \frac{1}{7}(0, 0, 1, 0, 1, 0, 0, 2, 1, 1, 1, 0)$

Inverse document frequency - idf

- Words that are very common in a corpus (that is occur in most documents) are often not informative. So the tf which measures only frequency in a single document has to be modulated by a specificity factor called idf. Broadly, it measures how often a word occurs across documents.

Let $D = \{D1, D2, D3, D4\} \equiv$corpus

$$idf(t, D) = ln \frac{|D|}{|\{d \in D | t \in D\}|}$$

Note that idf values are for a corpus. Given D the idf value for each term or word is fixed.

idf example

$D1$: `The cat sat on the mat. The cat was white.`
$D2$: `The brown coloured dog was barking loudly at the cat.`
$D3$: `The mat was green in colour.`
$D4$: `The dog pulled the mat with his teeth. The cat still sat on the mat.`
$V \equiv \{$`bark, brown, cat, colour, dog, green, loud, mat, pull, sit, teeth, white`$\}$, $|V| = 12$, $D = 4$.

idf values for each word are:
bark$=ln\frac{4}{1}$, brown$=ln\frac{4}{1}$, cat$=ln\frac{4}{3}$, colour$=ln\frac{4}{2}$, dog$=ln\frac{4}{2}$, green$=ln\frac{4}{1}$, loud$=ln\frac{4}{1}$, mat$=ln\frac{4}{3}$, pull$=ln\frac{4}{1}$, sit$=ln\frac{4}{2}$, teeth$=ln\frac{4}{1}$, white$=ln\frac{4}{1}$.

Term frequency inverse document frequency - tfidf

$$tfidf(t, d, D) = tf(t, d) \times idf(t, D)$$

For example the tfidf vector representation for $D1$ with normalized tf is:

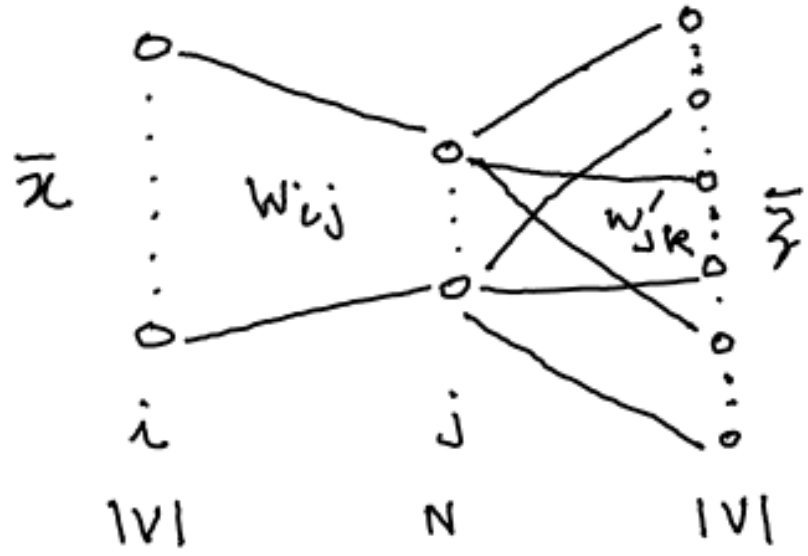
$\mathbf{d_1} = \frac{1}{5}(0, 0, 2.ln\frac{4}{3}, 0, 0, 0, 0, 1.ln\frac{4}{3}, 0, 1.ln\frac{4}{2}, 0, 1.ln\frac{4}{1})$

Word embeddings

- Word embedding is based on the distributional hypothesis which states 'a word is known by the company it keeps'. Another interpretation is words that occur in similar contexts have similar meanings.
- So, a word vector encodes the co-occurrence statistics between word and context for large text corpora in the form of a vector.
- There are multiple algorithms to compute word vectors. Some widely used recent ones are: Word2Vec: CBoW (Continuous bag of words), Skip-gram and GLoVE.

Skip-gram

Given word **x** as input predict C sized context around **x**.
Input **x**, true output vector **t** are one-hot encoded. **z** is the actual output. $|\mathbf{x}| = |\mathbf{t}| = |\mathbf{z}| = |V|$.



$W = [w_{ij}]_{|V| \times N}$ - embedding matrix (learnt), $W' = [w'_{jk}]_{N \times |V|}$ - output matrix, N - parameter, number of hidden units, embedding vector space dimension. **z** obtained by softmax.

Skip-gram training samples

Example: I rode on my cycle to the NLP class.

Let window size be 3 on either side of target word.

For target cycle the training pairs will be:

(cycle, rode), (cycle, on), (cycle, my), (cycle, to), (cycle, the), (cycle, NLP).

For target rode the training word pairs will be:

(rode, I), (rode, on), (rode, my), (rode, cycle).

Skip-gram forward pass

Let ω_x be the word whose one-hot representation is vector $\mathbf{x}$. Assume the index of ω_x in $\mathbf{x}$ is m, so $\mathbf{x}_m = 1$, $\mathbf{x}_i = 0, i \neq m$. Then the net input to the j^{th} unit in the hidden layer is:

$$net_j = \sum_{i=1}^{|V|} w_{ij}\mathbf{x}_i = w_{mj}\mathbf{x}_m = w_{mj}$$

If $\mathbf{h}$ is the hidden layer vector, $\mathbf{h} = \mathbf{w_m}$. $\mathbf{h}$ and $\mathbf{w_m}$ are $N-$dimensional vectors. There is no activation function for the hidden layer. So, $\mathbf{h}$ is actually the embedding vector of ω_x, $\mathbf{w_m}$. Net input for k^{th} unit in the output layer:

$$net_k = \sum_{j=1}^{N} \mathbf{w_m}_j w'_{jk}$$

The output z_k is computed as a *softmax*

$$z_k = \frac{e^{net_k}}{\sum_{l=1}^{|V|} e^{net_l}}$$

This will induce a probability distribution on $|V|$ units in the

Skip-gram backpropagation pass

The goal: maximize $p(z_{k^\star}|\mathbf{x})$ - that is probability for the correct output word whose index is assumed to be $k^\star$.

A possible error function: $-ln\, p(z_{k^\star}|\mathbf{x})$.

Use the error function and stochastic gradient descent to compute updates for both w'_{jk} and w_{ij}.

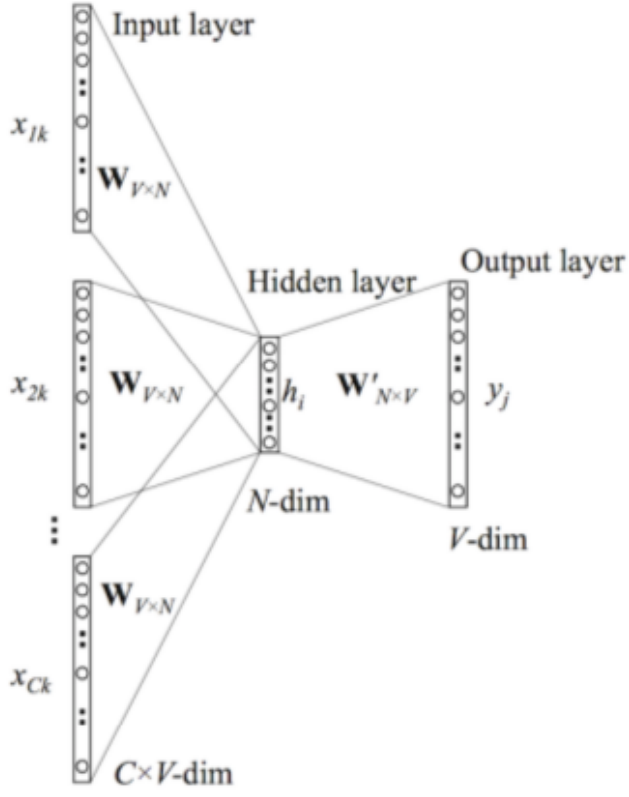
Versions of the skip-gram model that simultaneously predict C context words are also possible and lead to more complex update equations. The output weight matrix is shared between all context words.

Practical computation

- The skip-gram model involves computing a softmax over $|V|$ units. This is not viable in practice for large vocabularies and copora.
- Three main methods used to improve speed:
  - Treat common word pairs as single words.
  - Subsample frequent words to reduce number of training examples.
  - Approximate instead of doing full softmax. Multiple approaches are available - hierarchical softmax, sampling based
  - Use a technique called *negative sampling* to hugely decrease the number of weights updated in each round. Basically, select only a small number of 0 outputs for weight updating w.r.t to the true one-hot repn.

CBoW model

This is the reverse of the Skip-gram model. Input is the set of C context words and output is the target word.



(From: Minaar - tutorial)
w_{ij} are shared. $\mathbf{h} = \frac{1}{C} W . \sum_{l=1}^{C} \mathbf{x_l}$.

Properties of word vectors

- Analogy. Man:Woman::Uncle:Aunt, Man:Woman::King:Queen. Vector addition is able to solve analogies. Man-Woman+King*approx*Queen.
- Syntactic/morphological relation: Kings-King+Queen≈Queens.
- Relation examples.

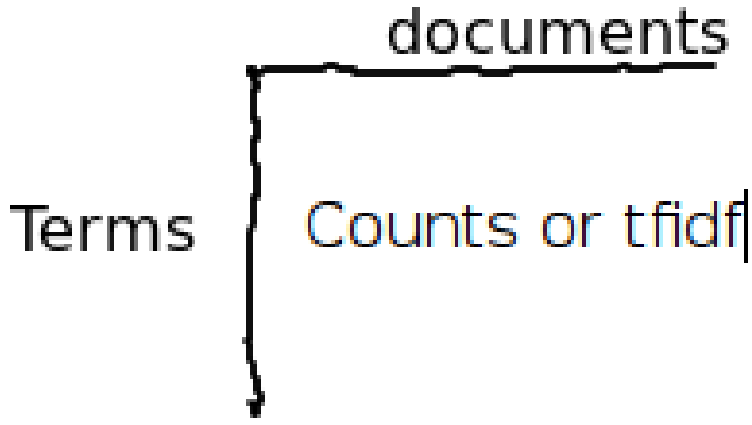
Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

More examples:

Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Latent semantic analysis

- Distributed semantic approach to extract latent concepts hidden in a set of documents using relationships between terms and documents.
- Term-document matrix - D.



Row vector i represents the realtion of the ith term with each document. Similarly, column vector j represents the relation between document j and all the terms.

- LSA finds a low rank approximation to the term-document matrix using *singular value decomposition* (SVD).

LSA uses

- The new low dimensional representation of documents (columns) can be used to classify documents using similarity (e.g. distance or cosine) and to cluster documents and/or terms.
- LSA can be used for information retrieval by representing the query or search string in the reduced dimensional space and then retrieve documents that are close or similar to the query representation.
- It can also be used to study word associations in a corpus - that is words that are close to or far from each other.

Finding low dimensional approx.

Let $D = [x_{ij}]_{m \times n}$ where m =no. of words, n =no. of documents. x_{ij} can be some kind of count, typically tfidf.

Using SVD D can be decomposed as:

$$D = U \Sigma V^T$$

where U and V are $m \times p$ and $p \times n$ orthogonal matrices and Σ is a $p \times p$ diagonal matrix containing singular values.

$$\begin{aligned} DD^T &= (U \Sigma V^T)(U \Sigma V^T)^T \\ &= U \Sigma V^T V \Sigma^T U^T \\ &= U \Sigma \Sigma^T U^T \\ &= U \Sigma^2 U^T \quad U \text{ contains eigenvectors of } DD^T \\ D^T D &= (U \Sigma V^T)^T (U \Sigma V^T) \\ &= V \Sigma^T U^T U \Sigma V^T \\ &= V \Sigma^2 V^T \quad V \text{ contains eigenvectors of } D^T D \end{aligned}$$

Both have same singular values Σ^2

Select k largest singular values and corresponding vectors from U and V. This gives a rank k approximation to D with the minimum Frœbenius norm.

$$|A| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{n} |a_{ij}|^2} = \sqrt{\sum_{i=1}^{min(m,n)} \sigma_i^2}$$

This gives a lower dimensional approximation to the term vectors and document vectors.

Corpus count based vectors - GLoVE

- HAL (Hyperspace Analogue to Language): Use a word-word matrix instead of the term-document matrix. Entries contain counts of target words occurring with context words. HAL does not handle frequently occurring words properly.
- LSA - does not actually produce word vectors. The term vectors do not have the nice properties that word vectors have.
- GLoVE: vector based on global co-ocurrence statistics. $X = [x_{ij}]$ where x_{ij} count of number of times term j occurs in the context of word i. $X_i = \sum_k X_{ik}$ count of any word occurring in context of word i. $P_{ij} = P(j|i) = \frac{x_{ij}}{X_i}$.

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

(From: Pennington et al, GLoVE)

Ratios like $\frac{P_{ik}}{P_{jk}}$ better capture difference between relevant word and irrelevant word.

So, desirable model is:

$$f(w_i, w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$$

, w_i, w_j are word vectors and $\tilde{w}_k$ is a context vector.

$f(w_i - w_j, \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$. Since right is a scalar choose
$f((w_i - w_j)^T \tilde{w}_k) = \frac{P_{ik}}{P_{jk}}$, in addition since target and context are symmetric so

$$f((w_i - w_j)^T \tilde{w}_k) = \frac{f(w_i^T \tilde{w}_k)}{f(w_j^T \tilde{w}_k)}$$

From earlier equation: $f(w_i^T \tilde{w}_k) = P_{ik} = \frac{x_{ik}}{X_i}$

Solution to above is: $log(P_{ik}) = log(x_{ik}) - log(X_i)$. To retain exchange symmetry absorb $log(X_i)$ (which is independent of k) into a bias b_i for X_i and add another bias $\tilde{b_k}$ for $\tilde{w}_k$ to restore symmetry. This gives the final form:

$$w_i^T \tilde{w}_k + b_i + \tilde{b_k} = log(X_{ik})$$

To remove divergence when $X_{ik} = 0$ use $log(1 + X_{ik})$.

Instead of factorizing the above matrix GLoVE solves a least square estimation problem by minimizing the error:

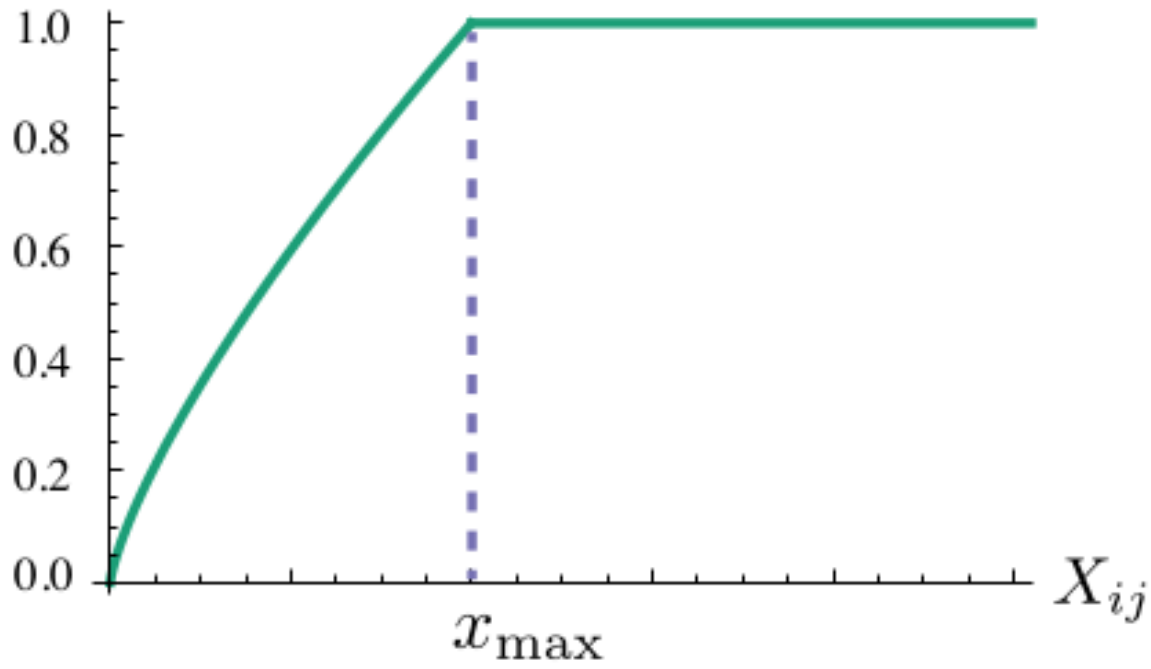
$$e = \sum_{i,j=1}^{|V|} g(X_{ij}(w_i^T \tilde{w}_j + b_i + \tilde{b}_j - log(X_{ij}))^2$$

g is a weighting function such that:

- $g(0) = 0$. g as a continuous function should vanish fast enough as $x \to 0$ so that $lim_{x \to 0} g(x) log^2 x$ is finite.
- $g(x)$ should be non-decreasing - rare co-occurrences are not overweighted.
- $g(x)$ should be small for large x so that frequent co-occurrences are modulated.

$$g(x) = \begin{cases} (\frac{x}{x_{max}})^{\alpha} & \text{if } x < x_{max} \\ 1 & \text{else} \end{cases}$$

α was chosen, empirically, as $\frac{3}{4}$. A plot of function g versus x and the value x_{max} is shown below.



Semantic similarity versus relatedness

- Word embeddings either through Word2Vec or GLoVE due to the algorithm satisfy a certain kind of semantic similarity - contextual similarity. Some call this semantic relatedness. For example consider GLoVE nearest vectors:

east	**expensive**	**British**
west	pricey	American
north	cheaper	Australian
south	costly	Britain
southeast	overpriced	European
northeast	inexpensive	England

- Are the nearest vectors *semantically similar*?
  Note that some of them are opposites or antonyms.

Modulating vectors with linguistic constraints[1]

- Existing vectors $\{\mathbf{v_1}, \ldots, \mathbf{v_n}\} \rightarrow \{\mathbf{v_1}', \ldots, \mathbf{v_n}'\}$ or $V \rightarrow V'$.
- Create set S and A of synonomous and antonymous pairs (i, j) of word vector indices in V.
- Define a 3-part objective function:

$$C(V, V') = c_1 AR(V') + c_2 SA(V') + c_3 VSP(V, V')$$

  where AR - antonym repel, SA - synonym attract, VSP - vector space preserve, c_is are constants.
- Run stochastic gradient descent to minimize $C(V, V')$ to get V'.
- Example of how to get task specific vectors (in this case for dialogue state tracking). One can create domain specific linguistic constraints (e.g. *(Indian,Chinese) as antonyms for restaurant dialogues*).

---

[1] N Mrskšić et al., Counter-fitting word vectors to linguistic constraints, arXiv:1603.00892v1, 2 Mar 2016

AR, SA, VSP

- $$AR(V') = \sum_{(u,w)\in A} \tau(\delta - d(\mathbf{v_u}', \mathbf{v_w}'))$$

  $d(v_i, v_j) = 1 - cos(v_i, v_j)$ - cosine similarity based distance, $\tau(x) \equiv max(0, x)$ imposes a margin on the cost, δ is the ideal distance for antonymous words (chosen as 1.0 - vector orthogonality).

- $$SA(V') = \sum_{(u,w)\in S} \tau(d(\mathbf{v_u}', \mathbf{v_w}') - \gamma)$$

  γ - ideal maximum distance between synonymous words (chosen as 0).

- $$VSP(V, V') = \sum_{i=1}^{n} \sum_{j\in Nbour(i)} \tau(d(\mathbf{v_i}', \mathbf{v_j}') - d(\mathbf{v_i}, \mathbf{v_j}))$$

Results of linguistic modulation

	east	**expensive**	**British**
Without modulation	west	pricey	American
	north	cheaper	Australian
	south	costly	Britain
	southeast	overpriced	European
	northeast	inexpensive	England
With modulation	eastward	costly	Brits
	eastern	pricy	London
	easterly	overpriced	BBC
	-	pricey	UK
	-	afford	Britain