LIMSI:
Learning Semantic Similarity by Selecting Random Word Subsets

Artem Sokolov
LIMSI-CNRS Orsay, France
gposhta@gmail.com

Highlights

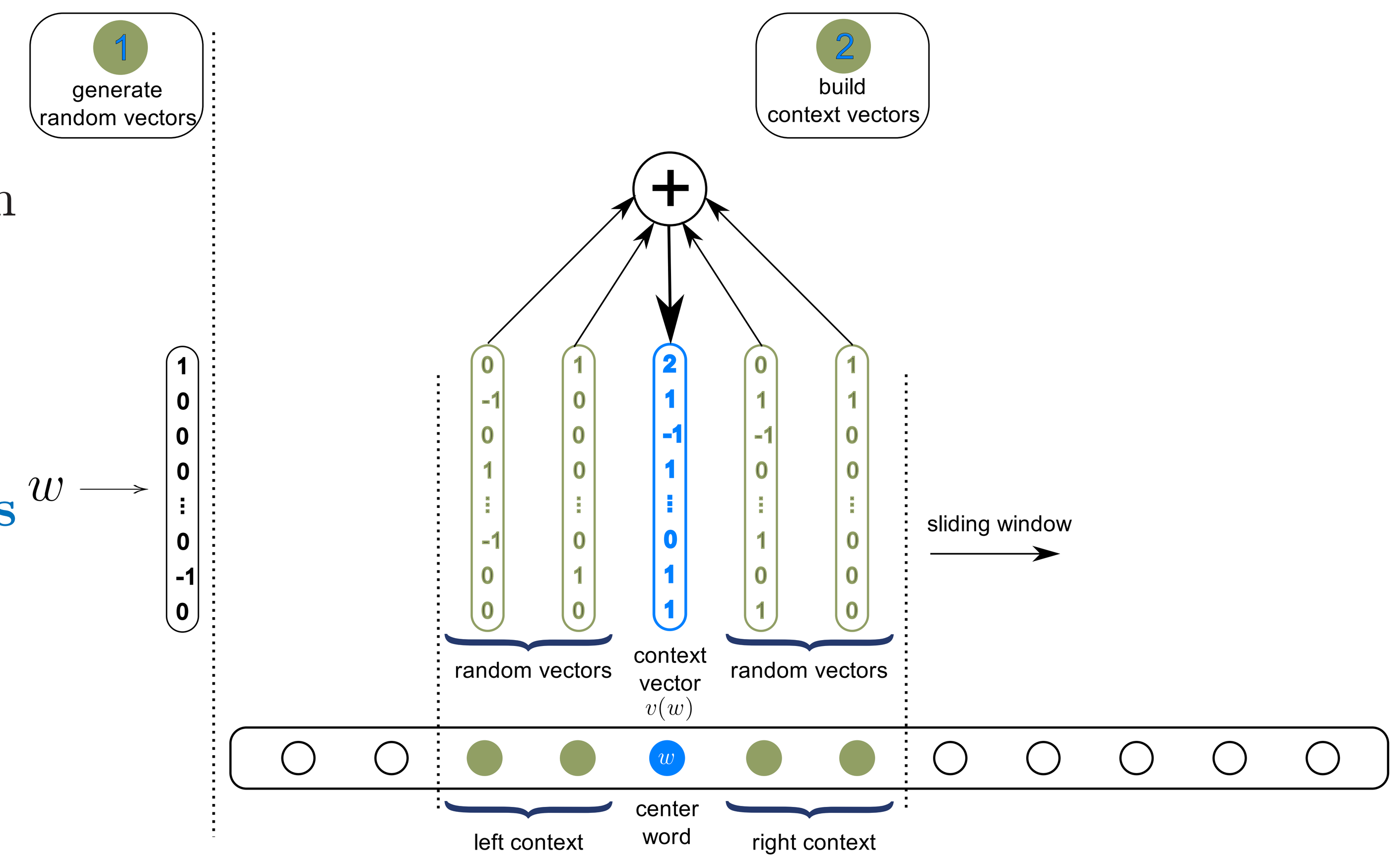
Semantic Textual Similarity learning between sentences

- **degrees of semantic equivalence:**
  5 (exactly the same meaning) .. → .. 0 (nothing in common)
- **task:**
  - given 2 sentences
  - predict the degree of semantic equivalence $\in [0, 5]$
  - evaluation: Pearson's correlation
- **contributions:**
  - ★ **improved Random Indexing (RI):**
    - ∗ selection of informative projection directions
    - ∗ learning-to-rank + boosting approach
  - ★ **no special preprocessing, no linguistic resources**
  - ★ **in top-25% among 89 participants** (SemEval'12/Task6)

Classic Random Indexing [Landauer and Dumais1997]

Distributional hypothesis: words with similar meanings tend to appear in similar contexts

- **map words w to context vectors $v(w)$:**
  - 1 initialize with ternary sparse vectors
  - 2 sum over contexts
  - 3-4 normalize for frequency and sentence length
  - ✔ similar contexts ⇒ similar vectors
- ★ **≃ dim. reduction by random projections**
  - ★ ℓ_2-norm between bags-of-words of contexts
  - ★ **ternarity turns on/off random word sets**
- **drawbacks:**
  - ✖ agnostic of the modeled semantic relation
  - ✖ trial-and-error tuning to specific task
  - ✖ all (even useless) random sets have effect
- **adaption to different tasks needed!**



3 normalize

$v(w) \leftarrow v(w)/\mathrm{freq}(w)$

4 sentence vectors

$v(\boldsymbol{s}) = \sum_{w\in \boldsymbol{s}} v(w)/|\boldsymbol{s}|$

RI setup

- no linguistic resources
- left/right context: 4 words
- dimension: 10^5
- initial non-zeros: 10
- vectors built on **GigaWord** (no punctuation, lowercased)

Symmetric Vector Transformations

Combine 2 RI sentence vectors $v(\boldsymbol{s}_1), v(\boldsymbol{s}_2)$ in a **symmetric** manner:

- 'sumdiff': $\bar{x} = (v(\boldsymbol{s}_1) + v(\boldsymbol{s}_2), sgn(v_1(\boldsymbol{s}_1) - v_1(\boldsymbol{s}_2))(v(\boldsymbol{s}_1) - v(\boldsymbol{s}_2)))$
- 'concat': $\bar{x} = (v(\boldsymbol{s}_1), v(\boldsymbol{s}_2))$, and $\bar{x}' = (v(\boldsymbol{s}_2), v(\boldsymbol{s}_1))$
- 'product': $x_i = v_i(\boldsymbol{s}_1) \cdot v_i(\boldsymbol{s}_2)$
- 'crossprod': $x_{ij} = v_i(\boldsymbol{s}_1) \cdot v_j(\boldsymbol{s}_2)$
- 'crossdiff': $x_{ij} = v_i(\boldsymbol{s}_1) - v_j(\boldsymbol{s}_2)$
- 'absdiff': $x_i = |v_i(\boldsymbol{s}_1) - v_i(\boldsymbol{s}_2)|$.

Performance on Training Data

	learner	transform	correlation	1σ
baseline	pure RI, cos	-	0.264	0.005
	logistic reg.	-	0.508	0.041
	logistic reg.	concat	0.537	0.052
boosting	RankBoost	**sumdiff**	0.685	0.027
		product	0.663	0.018
		crossprod	0.648	0.028
		crossdiff	0.643	0.023
		concat	0.625	0.025
		absdiff	0.602	0.021
	RtRank	**sumdiff**	0.730	0.020
		product	0.721	0.023

- selected transformations for the submission **in bold**.
- optimal parameters for RankBoost & RtRank found with cross-validation on 5 folds

Contrib.: Selecting Word Subsets with Boosting to Rank by Semantic Similarity

Idea:

- ★ exploit ordering of sentence pairs under y^i
- ★ use boosting to select important elements of $\bar{x}$ ⇔
  ⇔ select random word sets relevant to the similarity modeled

Given:

- training sentence pair: $\boldsymbol{s}_1^i, \boldsymbol{s}_2^i$ & similarity: y^i

Want to Learn:

- similarity score $H(\bar{x}) = \sum_{i=1}^{T} \alpha_t h_t(\bar{x})$, where
  - $\bar{x}$ – some symmetric transform of sentence vectors $v(\boldsymbol{s}_1), v(\boldsymbol{s}_2)$ (*see frame to the left*)
  - $h_t(\bar{x})$ – **simple (weak) functions** of $\bar{x}$, α_t – weights
  - the more similar $v(\boldsymbol{s}_1)$ is to $v(\boldsymbol{s}_2)$, the higher should be $H(\bar{x})$

Ranking losses & Weak functions h_t:

- **RankBoost** [Freund et al.2003]
  - loss: $\sum_{(\boldsymbol{s}_1^i,\boldsymbol{s}_2^i),(\boldsymbol{s}_1^j,\boldsymbol{s}_2^j): y^i<y^j} P(i,j)[\![H(\bar{x}^i) \geq H(\bar{x}^j)]\!]$
  - weak functions: $h(\boldsymbol{x}; \theta, k) = [\![x_k > \theta]\!]$ (decision stump)
  - weights $P(i,j) = y^j - y^i$
- **RtRank** [Zheng et al.2007, Mohan et al.2011]
  - loss: $\sum_{(\boldsymbol{s}_1^i,\boldsymbol{s}_2^i),(\boldsymbol{s}_1^j,\boldsymbol{s}_2^j): y^i<y^j} (\max\{0, H(\bar{x}^i) - H(\bar{x}^j)\})^2$
  - weak functions: decision trees of depth 4
- Well-known algorithms, implementations available online

Conclusion & Experimental Results

- ★ **RI extention:** selects word sets relevant for semantics learning
- ★ learns semantic order, not absolute values
- ★ **promising results**, despite ignoring preprocessing & linguistics

learner	transform	train±σ	test	final rank	MSRpar	MSRvid	SMTeur	OnWN	SMTnews
RankBoost	product	0.748±0.017	0.6392	32	0.3948	0.6597	0.0143	0.4157	0.2889
	sumdiff	0.735±0.016	0.6196	45	0.4295	0.5724	0.2842	0.3989	0.2575
RtRank	product	0.784±0.017	0.6789	22	0.4848	0.6636	0.0934	0.3706	0.2455
	sumdiff	0.763±0.014							

- underperforms on SMT sets: possible incorrect lexical choice in MT output + should be trained separately