MayoClinicNLP–CORE: Semantic representations for textual similarity

Stephen Wu
Mayo Clinic
Rochester, MN 55905
wu.stephen@mayo.edu

Dongqing Zhu & Ben Carterette
University of Delaware
Newark, DE 19716
{zhu,carteret}@cis.udel.edu

Hongfang Liu
Mayo Clinic
Rochester, MN 55905
liu.hongfang@mayo.edu

Abstract

The Semantic Textual Similarity (STS) task examines semantic similarity at a sentence-level. We explored three representations of semantics (implicit or explicit): named entities, semantic vectors, and structured vectorial semantics. From a DKPro baseline, we also performed feature selection and used source-specific linear regression models to combine our features. Our systems placed 5th, 6th, and 8th among 90 submitted systems.

1 Introduction

The Semantic Textual Similarity (STS) task (Agirre et al., 2012; Agirre et al., 2013) examines semantic similarity at a sentence-level. While much work has compared the semantics of terms, concepts, or documents, this space has been relatively unexplored. The 2013 STS task provided sentence pairs and a 0–5 human rating of their similarity, with training data from 5 sources and test data from 4 sources.

We sought to explore and evaluate the usefulness of several semantic representations that have had recent significance in research or practice. First, information extraction (IE) methods often implicitly consider named entities as ad hoc semantic representations, for example, in the clinical domain. Therefore, we sought to evaluate similarity based on named entity-based features. Second, in many applications, an effective means of incorporating distributional semantics is Random Indexing (RI). Thus we consider three different representations possible within Random Indexing (Kanerva et al., 2000; Sahlgren, 2005). Finally, because compositional distributional semantics is an important research topic (Mitchell and Lapata, 2008; Erk and Padó, 2008), we sought to evaluate a principled composition strategy: structured vectorial semantics (Wu and Schuler, 2011).

The remainder of this paper proceeds as follows. Section 2 overviews our similarity metrics, and Section 3 overviews the systems that were defined on these metrics. Competition results and additional analyses are in Section 4. We end with discussion on the results in Section 5.

2 Similarity measures

Because we expect semantic similarity to be multi-layered, we expect that we will need many similarity measures to approximate human similarity judgments. Rather than reinvent the wheel, we have chosen to introduce features that complement existing successful feature sets. We utilized 17 features from DKPro Similarity and 21 features from TakeLab, i.e., the two top-performing systems in the 2012 STS task, as a solid baseline.

These are summarized in Table 1. We introduce 3 categories of new similarity metrics, 9 metrics in all.

2.1 Named entity measures

Named entity recognition provides a common approximation of semantic content for the information extraction perspective. We define three simple similarity metrics based on named entities. First, we computed the *named entity overlap* (exact string matches) between the two sentences, where NE_k was the set of named entities found in sentence S_k. This is the harmonic mean of how closely $S1$

*Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task*, pages 148–154, Atlanta, Georgia, June 13-14, 2013.

Table 1: Full feature pool in MayoClinicNLP systems. The proposed MayoClinicNLP metrics are meant to complement DKPro (Bär et al., 2012) and TakeLab (Šarić et al., 2012) metrics.

DKPro metrics (17)	TakeLab metrics (21)	Custom MayoClinicNLP metrics (9)
n-grams/WordNGramContainmentMeasure_1_stopword-filtered	t_ngram/UnigramOverlap	
n-grams/WordNGramContainmentMeasure_2_stopword-filtered	t_ngram/BigramOverlap	
n-grams/WordNGramJaccardMeasure_1	t_ngram/TrigramOverlap	
n-grams/WordNGramJaccardMeasure_2_stopword-filtered	t_ngram/ContentUnigramOverlap	
n-grams/WordNGramJaccardMeasure_3	t_ngram/ContentBigramOverlap	
n-grams/WordNGramJaccardMeasure_4	t_ngram/ContentTrigramOverlap	
n-grams/WordNGramJaccardMeasure_4_stopword-filtered		
	t_words/WeightedWordOverlap	custom/StanfordNerMeasure_overlap.txt
	t_words/GreedyLemmaAligningOverlap	custom/StanfordNerMeasure_aligngst.txt
	t_words/WordNetAugmentedWordOverlap	custom/StanfordNerMeasure_alignlcs.txt
esa/ESA_Wiktionary	t_vec/LSAWordSimilarity_NYT	custom/SVSePhrSimilarityMeasure.txt
esa/ESA_WordNet	t_vec/LSAWordSimilarity_weighted_NYT	custom/SVSeTopSimilarityMeasure.txt
	t_vec/LSAWordSimilarity_weighted_Wiki	custom/SemanticVectorsSimilarityMeasure_d200_wr0.txt
		custom/SemanticVectorsSimilarityMeasure_d200_wr6b.txt
		custom/SemanticVectorsSimilarityMeasure_d200_wr6d.txt
		custom/SemanticVectorsSimilarityMeasure_d200_wr6p.txt
n-grams/CharacterNGramMeasure_2	t_other/RelativeLengthDifference	
n-grams/CharacterNGramMeasure_3	t_other/RelativeInfoContentDifference	
n-grams/CharacterNGramMeasure_4	t_other/NumbersSize	
string/GreedyStringTiling_3	t_other/NumbersOverlap	
string/LongestCommonSubsequenceComparator	t_other/NumbersSubset	
string/LongestCommonSubsequenceNormComparator	t_other/SentenceSize	
string/LongestCommonSubstringComparator	t_other/CaseMatches	
	t_other/StocksSize	
	t_other/StocksOverlap	

matches $S2$, and how closely $S2$ matches $S1$:

$$\mathrm{sim}_{neo}(S1, S2) = 2 \cdot \frac{|NE_1 \cap NE_2|}{|NE_1| + |NE_2|} \tag{1}$$

Additionally, we relax the constraint of requiring exact string matches between the two sentences by using the longest common subsequence (Allison and Dix, 1986) and greedy string tiling (Wise, 1996) algorithms. These metrics give similarities between two strings, rather than two sets of strings as we have with NE_1 and NE_2. Thus, we follow previous work in greedily aligning these named entities (Lavie and Denkowski, 2009; Šarić et al., 2012) into pairs. Namely, we compare each pair $(ne_{i,1}, ne_{j,2})$ of named entity strings in NE_1 and NE_2. The highest-scoring pair is entered into a set of pairs, P. Then, the next highest pair is added to P if neither named entity is already in P, and discarded otherwise; this continues until there are no more named entities in either NE_1 or NE_2.

We then define two named entity aligning measures that use the longest common subsequence (LCS) and greedy string tiling (GST) fuzzy string matching algorithms:

$$\mathrm{sim}_{nea}(S1, S2) = \frac{\sum\limits_{(ne_1, ne_2) \in P} f(ne_1, ne_2)}{\max\left(|NE_1|, |NE_2|\right)} \tag{2}$$

where $f(\cdot)$ is either the LCS or GST algorithm.

In our experiments, we performed named entity recognition with the Stanford NER tool using the standard English model (Finkel et al., 2005). Also, we used UKP's existing implementation of LCS and GST (Šarić et al., 2012) for the latter two measures.

2.2 Random indexing measures

Random indexing (Kanerva et al., 2000; Sahlgren, 2005) is another distributional semantics framework for representing terms as vectors. Similar to LSA (Deerwester et al., 1990), an index is created that represents each term as a semantic vector. But in random indexing, each term is represented by an elemental vector $\mathbf{e}_t$ with a small number of randomly-generated non-zero components. The intuition for this means of dimensionality reduction is that these randomly-generated elemental vectors are like quasi-orthogonal bases in a traditional geometric semantic space, rather than, e.g., 300 fully orthogonal dimensions from singular value decomposition (Landauer and Dumais, 1997). For a *standard model* with random indexing, a contextual term vector $\mathbf{c}_{t,\mathrm{std}}$ is the the sum of the elemental vectors corresponding to tokens in the document. All contexts for a particular term are summed and normalized to produce a final term vector $\mathbf{v}_{t,\mathrm{std}}$.

Other notions of context can be incorporated into this model. Local co-occurrence context can be accounted for in a *basic sliding-window model* by considering words within some window radius r (instead of a whole document). Each instance of the term t will have a contextual vector $\mathbf{c}_{t,\text{win}} = \mathbf{e}_{t-r} + \cdots + \mathbf{e}_{t-1} + \mathbf{e}_{t+1} + \cdots + \mathbf{e}_{t+r}$; context vectors for each instance (in a large corpus) would again be added and normalized to create the overall vector $\mathbf{v}_{t,\text{win}}$.

A *directional model* doubles the dimensionality of the vector and considers left- and right-context separately (half the indices for left-context, half for right-context), using a permutation to achieve one of the two contexts. A *permutated positional model* uses a position-specific permutation function to encode the relative word positions (rather than just left- or right-context) separately. Again, $\mathbf{v}_t$ would be summed and normalized over all instances of $\mathbf{c}_t$.

Sentence vectors from any of these 4 Random Indexing-based models (standard, windowed, directional, positional) are just the sum of the vectors for each term $\mathbf{v}_S = \sum_{t \in S} \mathbf{v}_t$. We define 4 separate similarity metrics for STS as:

$$\text{sim}_{RI}(S1, S2) = \cos(\mathbf{v}_{S1}, \mathbf{v}_{S2}) \tag{3}$$

We used the semantic vectors package (Widdows and Ferraro, 2008; Widdows and Cohen, 2010) in the default configuration for the standard model. For the windowed, directional, and positional models, we used a 6-word window radius with 200 dimensions and a seed length of 5. All models were trained on the raw text of the Penn Treebank Wall Street Journal corpus and a 100,075-article subset of Wikipedia.

2.3 Semantic vectorial semantics measures

Structured vectorial semantics (SVS) composes distributional semantic representations in syntactic context (Wu and Schuler, 2011). Similarity metrics defined with SVS inherently explore the qualities of a fully interactive syntax–semantics interface. While previous work evaluated the syntactic contributions of this model, the STS task allows us to evaluate the phrase-level semantic validity of the model. We summarize SVS here as bottom-up vector composition and parsing, then continue on to define the associated similarity metrics.

Each token in a sentence is modeled generatively as a vector $\mathbf{e}_\gamma$ of latent referents i_γ in syntactic context c_γ; each element in the vector is defined as:

$$\mathbf{e}_\gamma[i_\gamma] = \mathsf{P}(x_\gamma \mid lci_\gamma), \quad \text{for preterm } \gamma \tag{4}$$

where l_γ is a constant for preterminals.

We write SVS vector composition between two word (or phrase) vectors in linear algebra form,[1] assuming that we are composing the semantics of two children $\mathbf{e}_\alpha$ and $\mathbf{e}_\beta$ in a binary syntactic tree into their parent $\mathbf{e}_\gamma$:

$$\mathbf{e}_\gamma = \mathbf{M} \odot (\mathbf{L}_{\gamma\times\alpha} \cdot \mathbf{e}_\alpha) \odot (\mathbf{L}_{\gamma\times\beta} \cdot \mathbf{e}_\beta) \cdot \mathbf{1} \tag{5}$$

$\mathbf{M}$ is a diagonal matrix that encapsulates probabilistic syntactic information; the $\mathbf{L}$ matrices are linear transformations that capture how semantically relevant child vectors are to the resulting vector (e.g., $\mathbf{L}_{\gamma\times\alpha}$ defines the the relevance of $\mathbf{e}_\alpha$ to $\mathbf{e}_\gamma$). These matrices are defined such that the resulting $\mathbf{e}_\gamma$ is a semantic vector of consistent $\mathsf{P}(x_\gamma \mid lci_\gamma)$ probabilities. Further detail is in our previous work (Wu, 2010; Wu and Schuler, 2011).

Similarity metrics can be defined in the SVS space by comparing the distributions of the composed $\mathbf{e}_\gamma$ vectors — i.e., our similarity metric is a comparison of the vector semantics at different phrasal nodes. We define two measures, one corresponding to the top node $c_\triangle$ (e.g., with a syntactic constituent $c_\triangle$ = 'S'), and one corresponding to the left and right largest child nodes (e.g.,, $c_\angle$ = 'NP' and $c_\measuredangle$ = 'VP' for a canonical subject–verb–object sentence in English).

$$\text{sim}_{svs\text{-}top}(S1, S2) = \cos(\mathbf{e}_{\triangle(S1)}, \mathbf{e}_{\triangle(S2)}) \tag{6}$$

$$\begin{aligned}\text{sim}_{svs\text{-}phr}(S1, S2) = \max(&\\ &\text{avgsim}(\mathbf{e}_{\angle(S1)}, \mathbf{e}_{\angle(S2)}; \mathbf{e}_{\measuredangle(S1)}, \mathbf{e}_{\measuredangle(S2)}),\\ &\text{avgsim}(\mathbf{e}_{\angle(S1)}, \mathbf{e}_{\measuredangle(S2)}; \mathbf{e}_{\measuredangle(S1)}, \mathbf{e}_{\angle(S2)}))\end{aligned} \tag{7}$$

where avgsim() is the harmonic mean of the cosine similarities between the two pairs of arguments. Top-level similarity comparisons in (6) amounts to comparing the semantics of a whole sentence. The phrasal similarity function $\text{sim}_{svs\text{-}phr}(S1, S2)$ in (7) thus seeks to semantically align the two largest subtrees, and weight them. Compared to $\text{sim}_{svs\text{-}top}$,

[1] We define the operator $\odot$ as point-by-point multiplication of two diagonal matrices and $\mathbf{1}$ as a column vector of ones, collapsing a diagonal matrix onto a column vector.

the phrasal similarity function $\text{sim}_{svs\text{-}phr}(S1, S2)$ assumes there might be some information captured in the child nodes that could be lost in the final composition to the top node.

In our experiments, we used the parser described in Wu and Schuler (2011) with 1,000 headwords and 10 relational clusters, trained on the Wall Street Journal treebank.

3 Feature combination framework

The similarity metrics of Section 2 were calculated for each of the sentence pairs in the training set, and later the test set. In combining these metrics, we extended a DKPro Similarity baseline (3.1) with feature selection (3.2) and source-specific models and classification (3.3).

3.1 Linear regression via DKPro Similarity

For our baseline (MayoClinicNLPr1wtCDT), we used the UIMA-based DKPro Similarity system from STS 2012 (Bär et al., 2012). Aside from the large number of sound similarity measures, this provided linear regression through the WEKA package (Hall et al., 2009) to combine all of the disparate similarity metrics into a single one, and some preprocessing. Regression weights were determined on the whole training set for each source.

3.2 Feature selection

Not every feature was included in the final linear regression models. To determine the best of the 47 (DKPro–17, TakeLab–21, MayoClinicNLP–9) features, we performed a full forward-search on the space of similarity measures. In forward-search, we perform 10-fold cross-validation on the training set for each measure, and pick the best one; in the next round, that best metric is retained, and the remaining metrics are considered for addition. Rounds continue until all the features are exhausted, though a stopping-point is noted when performance no longer increases.

3.3 Subdomain source models and classification

There were 5 sources of data in the training set: paraphrase sentence pairs (MSRpar), sentence pairs from video descriptions (MSRvid), MT evaluation sentence pairs (MTnews and MTeuroparl) and gloss pairs (OnWN). In our submitted runs, we trained a separate, feature-selected model based on cross-validation for each of these data sources. In training data on cross-validation tests, training domain-specific models outperformed training a single conglomerate model.

In the test data, there were 4 sources, with 2 appearing in training data (OnWN, SMT) and 2 that were novel (FrameNet/Wordnet sense definitions (FNWN), European news headlines (headlines)). We examined two different strategies for applying the 5-source trained models on these 4 test sets. Both of these strategies rely on a multiclass random forest *classifier*, which we trained on the 47 similarity metrics.

First, for each sentence pair, we considered the final similarity score to be a weighted combination of the similarity score from each of the 5 source-specific similarity models. The combination weights were determined by utilizing the classifier's confidence scores. Second, the final similarity was chosen as the single source-specific similarity score corresponding to the classifier's output class.

4 Evaluation

The MayoClinicNLP team submitted three systems to the STS-Core task. We also include here a post-hoc run that was considered as a possible submission.

r1wtCDT This run used the 47 metrics from DKPro, TakeLab, and MayoClinicNLP as a feature pool for feature selection. Source-specific similarity metrics were combined with classifier-confidence-score weights.

r2CDT Same feature pool as run 1. Best-match (as determined by classifier) source-specific similarity metric was used rather than a weighted combination.

r3wtCD TakeLab features were removed from the feature pool (before feature selection). Same source combination as run 1.

r4ALL Post-hoc run using all 47 metrics, but training a single linear regression model rather than source-specific models.

Table 2: Performance comparison.

TEAM NAME	headlines	rank	OnWN	rank	FNWN	rank	SMT	rank	mean	rank
UMBC_EBIQUITY-ParingWords	0.7642		0.7529		0.5818		0.3804		0.6181	1
UMBC_EBIQUITY-galactus	0.7428		0.7053		0.5444		0.3705		0.5927	2
deft-baseline	0.6532		0.8431		0.5083		0.3265		0.5795	3
MayoClinicNLP-r4ALL	0.7275		0.7618		0.4359		0.3048		0.5707	
UMBC_EBIQUITY-saiyan	0.7838		0.5593		0.5815		0.3563		0.5683	4
MayoClinicNLP-r3wtCD	0.6440	43	0.8295	2	0.3202	47	0.3561	17	0.5671	5
MayoClinicNLP-r1wtCDT	0.6584	33	0.7775	4	0.3735	26	0.3605	13	0.5649	6
CLaC-RUN2	0.6921		0.7366		0.3793		0.3375		0.5587	7
MayoClinicNLP-r2CDT	0.6827	23	0.6612	20	0.396	17	0.3946	5	0.5572	8
NTNU-RUN1	0.7279		0.5952		0.3215		0.4015		0.5519	9
CLaC-RUN1	0.6774		0.7667		0.3793		0.3068		0.5511	10

4.1 Competition performance

Table 2 shows the top 10 runs of 90 submitted in the STS-Core task are shown, with our three systems placing 5th, 6th, and 8th. Additionally, we can see that run 4 would have placed 4th. Notice that there are significant source-specific differences between the runs. For example, while run 4 is better overall, runs 1–3 outperform it on all but the headlines and FNWN datasets, i.e., the test datasets that were not present in the training data. Thus, it is clear that the source-specific models are beneficial when the training data is in-domain, but a combined model is more beneficial when no such training data is available.

4.2 Feature selection analysis

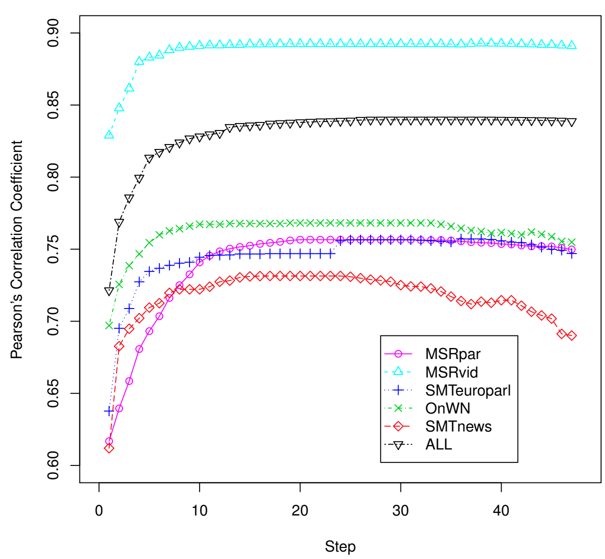

Figure 1: Performance curve of feature selection for **r1wtCDT**, **r2CDT**, and **r4ALL**

Due to the source-specific variability among the runs, it is important to know whether the forward-search feature selection performed as expected. For source specific models (runs 1 and 3) and a combined model (run 4), Figure 1 shows the 10-fold cross-validation scores on the training set as the next feature is added to the model. As we would expect, there is an initial growth region where the first features truly complement one another and improve performance significantly. A plateau is reached for each of the models, and some (e.g., SMTnews) even decay if too many noisy features are added.

The feature selection curves are as expected. Because the plateau regions are large, feature selection could be cut off at about 10 features, with gains in efficiency and perhaps little effect on accuracy.

The resulting selected features for some of the trained models are shown in Table 3.

4.3 Contribution of MayoClinicNLP metrics

We determined whether including MayoClinicNLP features was any benefit over a feature-selected DKPro baseline. Table 4 analyzes this question by adding each of our measures in turn to a baseline feature-selected DKPro (dkselected). Note that this baseline was extremely effective; it would have ranked 4th in the STS competition, outperforming our run 4. Thus, metrics that improve this baseline must truly be complementary metrics. Here, we see that only the phrasal SVS measure is able to improve performance overall, largely by its contributions to the most difficult categories, FNWN and SMT. In fact, that system (dkselected + SVSePhrSimilarityMeasure) represents the best-performing run of any that was produced in our framework.

Table 3: Top retained features for several linear regression models.

OnWN - r1wtCDT and r2CDT (15 shown/19 selected)	SMTnews - r1wtCDT and r2CDT (15 shown/17 selected)	All - r4ALL (29 shown/29 selected)
t_ngram/ContentUnigramOverlap	t_other/RelativeInfoContentDifference	t_vec/LSAWordSimilarity_weighted_NYT
t_other/RelativeInfoContentDifference	n-grams/CharacterNGramMeasure_2	n-grams/CharacterNGramMeasure_2
t_vec/LSAWordSimilarity_weighted_NYT	t_other/CaseMatches	string/LongestCommonSubstringComparator
esa/ESA_Wiktionary	string/GreedyStringTiling_3	t_other/NumbersOverlap
t_ngram/ContentBigramOverlap	**custom/RandomIndexingMeasure_d200_wr6p**	t_words/WordNetAugmentedWordOverlap
n-grams/CharacterNGramMeasure_2	**custom/StanfordNerMeasure_overlap**	n-grams/WordNGramJaccardMeasure_1
t_words/WordNetAugmentedWordOverlap	t_vec/LSAWordSimilarity_weighted_NYT	n-grams/CharacterNGramMeasure_3
t_ngram/BigramOverlap	t_other/SentenceSize	t_other/SentenceSize
string/GreedyStringTiling_3	**custom/RandomIndexingMeasure_d200_wr0**	t_other/RelativeInfoContentDifference
string/LongestCommonSubsequenceNormComparator	**custom/SVSePhrSimilarityMeasure**	t_ngram/ContentBigramOverlap
custom/RandomIndexingMeasure_d200_wr0	esa/ESA_Wiktionary	n-grams/WordNGramJaccardMeasure_4
custom/StanfordNerMeasure_aligngst	string/LongestCommonSubstringComparator	t_other/NumbersSize
custom/StanfordNerMeasure_alignlcs	t_other/NumbersSize	t_other/NumbersSubset
custom/StanfordNerMeasure_overlap	n-grams/WordNGramContainmentMeasure_2_stopword-filtered	**custom/SVSePhrSimilarityMeasure**
custom/SVSePhrSimilarityMeasure	**custom/SVSeTopSimilarityMeasure**	**custom/SemanticVectorsSimilarityMeasure_d200_wr6p**
		esa/ESA_WordNet
OnWN - r3wtCD (7 shown/7 selected)	SMTnews - r3wtCD (15 shown/23 selected)	esa/ESA_Wiktionary
esa/ESA_Wiktionary	string/GreedyStringTiling_3	string/LongestCommonSubsequenceComparator
string/LongestCommonSubsequenceComparator	**custom/StanfordNerMeasure_overlap**	string/LongestCommonSubsequenceNormComparator
string/GreedyStringTiling_3	n-grams/CharacterNGramMeasure_2	n-grams/WordNGramContainmentMeasure_1_stopword-filtered
string/LongestCommonSubsequenceNormComparator	**custom/RandomIndexingMeasure_d200_wr6p**	word-sim/MCS06_Resnik_WordNet
string/LongestCommonSubstringComparator	n-grams/CharacterNGramMeasure_3	t_ngram/ContentUnigramOverlap
word-sim/MCS06_Resnik_WordNet	string/LongestCommonSubsequenceComparator	n-grams/WordNGramContainmentMeasure_2_stopword-filtered
n-grams/WordNGramContainmentMeasure_2_stopword-filtered	**custom/StanfordNerMeasure_aligngst**	n-grams/WordNGramJaccardMeasure_2_stopword-filtered
	custom/SVSePhrSimilarityMeasure	t_ngram/UnigramOverlap
	esa/ESA_Wiktionary	t_ngram/BigramOverlap
	esa/ESA_WordNet	t_other/StocksSize
	n-grams/WordNGramContainmentMeasure_2_stopword-filtered	t_words/GreedyLemmaAligningOverlap
	n-grams/WordNGramJaccardMeasure_1	t_other/StocksOverlap
	string/LongestCommonSubstringComparator	
	custom/RandomIndexingMeasure_d200_wr6d	
	custom/RandomIndexingMeasure_d200_wr0	

Table 4: Adding customized features one at a time into optimized DKPro feature set. Models are trained across all sources.

	headlines	OnWN	FNWN	SMT	mean
dkselected	0.70331	0.79752	0.38358	0.31744	0.571319
dkselected + SVSePhrSimilarityMeasure	0.70178	0.79644	**0.38685**	**0.32332**	**0.572774**
dkselected + RandomIndexingMeasure_d200_wr0	0.70054	0.79752	**0.38432**	0.31615	0.570028
dkselected + SVSeTopSimilarityMeasure	0.69873	0.79522	**0.38815**	0.31723	0.569533
dkselected + RandomIndexingMeasure_d200_wr6d	0.69944	**0.79836**	**0.38416**	0.31397	0.569131
dkselected + RandomIndexingMeasure_d200_wr6b	0.69992	**0.79788**	**0.38435**	0.31328	0.568957
dkselected + RandomIndexingMeasure_d200_wr6p	0.69878	**0.79848**	0.37876	0.31436	0.568617
dkselected + StanfordNerMeasure_aligngst	0.69446	0.79502	**0.38703**	0.31497	0.567212
dkselected + StanfordNerMeasure_overlap	0.69468	0.79509	**0.38703**	0.31466	0.567200
dkselected + StanfordNerMeasure_alignlcs	0.69451	0.79486	**0.38657**	0.31394	0.566807
(dk + all custom) selected	0.70311	0.79887	0.37477	0.31665	0.570586

Also, we see some source-specific behavior. None of our introduced measures are able to improve the headlines similarities. However, random indexing improves OnWN scores, several strategies improve the FNWN metric, and $\mathrm{sim}_{\textit{svs-phr}}$ is the only viable performance improvement on the SMT corpus.

5 Discussion

Mayo Clinic's submissions to Semantic Textual Similarity 2013 performed well, placing 5th, 6th, and 8th among 90 submitted systems. We introduced similarity metrics that used different means to do compositional distributional semantics along with some named entity-based measures, finding some improvement especially for phrasal similarity from structured vectorial semantics. Throughout, we utilized forward-search feature selection, which enhanced the performance of the models. We also used source-based linear regression models and considered unseen sources as mixtures of existing sources; we found that in-domain data is necessary for smaller, source-based models to outperform larger, conglomerate models.

Acknowledgments

Thanks to the developers of the UKP DKPro system and the TakeLab system for making their code available. Also, thanks to James Masanz for initial implementations of some similarity measures.

References

Eneko Agirre, Mona Diab, Daniel Cer, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 385–393. Association for Computational Linguistics.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity, including a pilot on typed-similarity. In **SEM 2013: The Second Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics.

Lloyd Allison and Trevor I Dix. 1986. A bit-string longest-common-subsequence algorithm. *Information Processing Letters*, 23(5):305–310.

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 435–440. Association for Computational Linguistics.

Scott Deerwester, Susan Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP 2008*.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370. Association for Computational Linguistics.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11(1):10–18, November.

Pentti Kanerva, Jan Kristofersson, and Anders Holst. 2000. Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd annual conference of the cognitive science society*, volume 1036. Citeseer.

T.K. Landauer and S.T. Dumais. 1997. A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review*, 104:211–240.

Alon Lavie and Michael J Denkowski. 2009. The meteor metric for automatic evaluation of machine translation. *Machine translation*, 23(2-3):105–115.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, OH.

M. Sahlgren. 2005. An introduction to random indexing. In *Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE*, volume 5.

Frane Šarić, Goran Glavaš, Mladen Karan, Jan Šnajder, and Bojana Dalbelo Bašić. 2012. Takelab: Systems for measuring semantic text similarity. In *Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 441–448, Montréal, Canada, 7-8 June. Association for Computational Linguistics.

Dominic Widdows and Trevor Cohen. 2010. The semantic vectors package: New algorithms and public tools for distributional semantics. In *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, pages 9–15. IEEE.

D. Widdows and K. Ferraro. 2008. Semantic vectors: a scalable open source package and online technology management application. *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, pages 1183–1190.

Michael J Wise. 1996. Yap3: Improved detection of similarities in computer program and other texts. In *ACM SIGCSE Bulletin*, volume 28, pages 130–134. ACM.

Stephen Wu and William Schuler. 2011. Structured composition of semantic vectors. In *Proceedings of the International Conference on Computational Semantics*.

Stephen Tze-Inn Wu. 2010. *Vectorial Representations of Meaning for a Computational Model of Language Comprehension*. Ph.D. thesis, Department of Computer Science and Engineering, University of Minnesota.