SATTY : Word Sense Induction Application in Web Search Clustering*

Satyabrata Behera
IIT Bombay
Mumbai, India
satty@cse.iitb.ac.in

Upasana Gaikwad
IIT Bombay
Mumbai, India
upasana@cse.iitb.ac.in

Ramakrishna Bairi
IIT Bombay
Mumbai, India
bairi@cse.iitb.ac.in

Ganesh Ramakrishnan
IIT Bombay
Mumbai, India
ganesh@cse.iitb.ac.in

Abstract

The aim of this paper is to perform Word Sense induction (WSI); which clusters web search results and produces a diversified list of search results. It describes the WSI system developed for Task 11 of SemEval - 2013. This paper implements the idea of *monotone submodular* function optimization using greedy algorithm.

1 Introduction

Two different types of systems were submitted under Task 11 of SemEval - 2013 (Roberto Navigli and Daniele Vannella, 2013). The two system types are WSI (Word Sense Induction) and WSD (Word Sense Disambiguation). WSD is the task of automatically associating meaning with words. In WSD the possible meanings for a given word are drawn from an existing sense inventory. In contrast, WSI aims at automatically identifying the meanings of a given word from raw text. A WSI system will be asked to identify the meaning of the input query and cluster the search results into semantically-related groups according to their meanings. Instead, a WSD system will be requested to sense-tag the above search results with the appropriate senses of the input query and this, again, will implicitly determine a clustering of snippets (i.e., one cluster per sense).

*This system was designed and submitted in the competition SemEval-2013 under task 11 : Evaluating Word Sense Induction & Disambiguation within An End-User Application (Roberto Navigli and Daniele Vannella2013). http://www.cs.york.ac.uk/semeval-2013/.

Our system implements the idea given in (Jingrui He and Hanghang Tong and Qiaozhu Mei and Boleslaw Szymanski, 2012). This developed system uses the concept of submodularity. The task is treated as a submodular function maximization which has its benefits. On the one hand, there exists a simple greedy algorithm for monotone submodular function maximization where the solution obtained is guaranteed to be almost as good as the best possible solution according to an objective. More precisely, the greedy algorithm is a constant factor approximation to the cardinality constrained version of the problem, so that the approximate soultion is in the bound of $(1 - 1/e)$ of optimal solution. It is also important to note that this is a worst case bound, and in most cases the quality of the solution obtained will be much better than this bound suggests. In our system, monotone submodular objective of (Jingrui He and Hanghang Tong and Qiaozhu Mei and Boleslaw Szymanski, 2012) was implemented to find the top k simultaneously relevant and diversified list of search results. Once these top k results are obtained, they are used as centroids to form clusters by classifying each of remaining search results to one of the centroid with maximum similarity, producing k clusters. Those results which are not similar to any of the centroids are either put in a different cluster or are assigned to cluster with highest similarity.

2 Background on Submodularity

Our system uses the concept of submodularity. Given a set of objects $V = v_1, ..., v_n$ and a function $F : 2^V \rightarrow \mathbf{R}$ that returns a real value for any subset

$S \subseteq V$. The function F is said to be submodular if it satisfies the property of *diminishing returns*, i.e., $A \subseteq B \subseteq V \setminus v$, a submodular function F must satisfy $F(A + v) - F(A) \geq F(B + v) - F(B)$. A set function F is *monotone nondecreasing* if $\forall A \subseteq B, F(A) \geq F(B)$. A *monotone nondecreasing* submodular function is referred to as monotone submodular.

We need to find the subset of bounded size $|S| \leq k$ that maximizes the function F, e.g. $argmax_{S \subseteq V} F(S)$. In general, this operation is intractable. As shown in (G.L. Nemhauser and L.A. Wolsey, 1978), if function F is monotone submodular, then a simple greedy algorithm finds an approximate solution which is guaranteed to be within $(1 - 1/e) \sim 0.63$ of optimal solution. Many properties of submodular functions are common with convex and concave functions (L. Lovász, 1983). One of those is that they are closed under a number of common combination operations such as summation, certain compositions, restrictions etc.

Previous work on submodularity is in (Hui Lin and Jeff Bilmes, 2011) where a monotone submodular objective is maximized using a greedy algorithm for document summarization. The objective function is:

$$F(S) = L(S) + \lambda R(S)$$

where $L(S)$ measures the coverage of summary set S to the document V and $R(S)$ measures diversity in S, which are properties of a good summary. $\lambda \geq 0$ is trade-off coefficient. V represents all the sentences (or other linguistic units) in a document (or document collection). Also $L(S)$ and $R(S)$ are monotone submodular functions. This work was again extended in (Hui Lin and Jeff A. Bilmes, 2012) where the submodular objective is itself a weighted combination of several submodular functions, where the weights are learnt in a max-margin setting. This work also demonstrates the use of this idea for document summarization.

3 System Description

The system works in 2 stages:

1. The first stage produces top k diversified and relevant set of search results.
2. The second stage forms k clusters of search results treating top k results as centroids.

The problem of finding top k diversified and relevant search results is posed as an optimization problem. This optimization function has the property of diminishing returns and monotonicity, which is a **monotone submodular function**. This enables to design a scalable, greedy algorithm to find the $(1 - 1/e)$ near-optimal solution. The optimization function is taken from (Jingrui He and Hanghang Tong and Qiaozhu Mei and Boleslaw Szymanski, 2012) and presented below.

Objective Function : The aim is to find a subset T of k search results which optimizes the objective function.

$$argmax_{|T|=k} \; w \sum_{i \in T} q_i r_i - \sum_{i,j \in T} r_i S_{ij} r_j$$

where, T is the subset of search results. $q = S.r$ is a nx1 vector. Intuitively, its i^{th} element q_i measures the importance of i^{th} search result. To be specific, if x_i is similar to many search results that are highly relevant to the query, it is more important than the search results whose neighbours are less relevant. S is a nxn similarity matrix between search results. r is a nx1 relevance vector of search results to query. w is a regularization parameter which defines trade-off between two terms.

The first term of the objective function measures the total weighted relevance of T with respect to query. It favours relevant search results from big clusters. In other words, if two search results are equally relevant to the query, one from a big cluster and the other isolated, by using weighted relevance, it prefers the former.

The second term measures the similarity among the search results within T such that it penalizes the selection of multiple relevant search results that are very similar to each other. By including this term in the objective function, we try to find a set of search results which are highly relevant to the query and also dissimilar to each other.

As the objective function is monotone submodular, the greedy algorithm finds the top k search re-

sults (i.e. near optimal solution) with approximation guarantee of $(1 - 1/e)$.

The second stage performs clustering using results of previous stage. The top k search results output by the previous stage are treated as centroids and the remaining search results are assigned to the centroid with the maximum similarity.

4 Experimental Results

The implemented system was tested on data given by SemEval - 2013[1](Roberto Navigli and Daniele Vannella, 2013). Data contains 100 queries, each with 64 search results. Each search result contains title, url and snippet.

Only title and snippet information was used. The relevance between query and a search result is calculated using weighted Jaccard. Cosine similarity is used to calculate the similarity between search results using only title and snippet. It was just bag of words (i.e. unigram) approach and no other preprocessing of data was done. In the first stage, system produces top 10 diversified search results which are then used as centroids to form 10 clusters. Those results which are not similar to any of the centroids are put in a different cluster, sometimes resulting in 11 clusters.

The evaluation method required : (i) to rank the search results within each cluster according to the confidence with which they belong to that cluster, (ii) to rank the clusters according to their diversity.

The cluster ranking is kept same as the rank of their centroids in top 10 results returned in first stage of the system.

Also search results within each cluster are then ranked by their average similarity to rest of the search results in the same cluster, in descending order with respect to the ranking score. The ranking score of search result x_i in cluster C is calculated as below, which is used in our system :

$$score(x_i) = \frac{1}{|C| - 1} \sum_{j:j \in C, i \neq j} S_{ij}$$

[1]http://www.cs.york.ac.uk/semeval-2013/task11/index.php?id=data

The other way of ranking search results within a cluster can be ranking by their relevance to the query. In that case, it depends on how good the relevance scores are. This ranking affects the ability of the system to diversify search results, i.e., *Subtopic Recall@K* and *Subtopic Precision@r* measures. The clustering quality is measured by measures of Rand Index (RI), Adjusted Rand Index (ARI), F1-measure (F1) and Jaccard Index (JI). All these evaluation metrics used are described in (Antonio Di Marco and Roberto Navigli, 2013). All the given evaluation metric values are obtained for the described data using the java evaluator provided by SemEval - 2013 (Roberto Navigli and Daniele Vannella, 2013). Our system's evaluation measures along with other systems, submitted in SemEval - 2013 are shown in tables 1, 2 and 3. Our system's name is task11-satty-approach1.

The clustering quality was found to be good as indicated by F1 and RI while scoring low for ARI, JI. In terms of diversification of search results, it did not perform that well indicating that either ranking of search results within each cluster or cluster ranking or both were not that good.

5 Conclusion

In this paper Word Sense Induction was implemented on web search clustering. The developed system evaluated with respect to different evaluation metrics. The system's clustering quality was found to be good while its ability to diversify search results was not that good. Better ranking of clusters as well as ranking of search results within each cluster can improve the system's ability to diversify search results.

The similarity score between search results were calculated using only title and snippet, but it can also be evaluated by fetching whole document. Since the relevance score of each search result to the query was not available, it was calculated by considering occurrence frequency of query words in search results (i.e. title and snippet). If a better relevance score were available by the search engine, the system might have performed better. These two aspects can be tested in further work.

System	Type	F1	ARI	RI	Jaccard	Avg. No. of Clusters	Avg. Cluster Size
hdp-clusters-lemma	WSI	**0.683**	0.2131	**0.6522**	0.3302	6.63	11.0756
hdp-clusters-nolemma	WSI	0.6803	**0.2149**	0.6486	0.3375	6.54	11.6803
task11-satty-approach1	WSI	0.6709	0.0719	0.5955	0.1505	9.9	6.4631
task11-ukp-wsi-wp-pmi	WSI	0.6048	0.0364	0.505	0.2932	5.86	30.3098
task11.duluth.sys7.pk2	WSI	0.5878	0.0678	0.5204	0.3103	3.01	25.1596
task11-ukp-wsi-wp-llr2	WSI	0.5864	0.0377	0.5109	0.3177	4.17	21.8702
task11-ukp-wsi-wacky-llr	WSI	0.5826	0.0253	0.5002	**0.3394**	3.64	32.3434
task11.duluth.sys9.pk2	WSI	0.5702	0.0259	0.5463	0.2224	3.32	19.84
task11.duluth.sys1.pk2	WSI	0.5683	0.0574	0.5218	0.3179	2.53	26.4533
rakesh	WSD	0.3949	0.0811	0.5876	0.3052	9.07	2.9441
singleton		1.0000	0.0000	0.6009	0.0000	64.0000	1.0000
allinone		0.5442	0.0000	0.3990	0.3990	1.0000	64.0000
gold		1.0000	0.9900	1.0000	1.0000	7.6900	11.5630

Table 1: The best result for each column is presented in boldface. **singleton** and **allinone** are baseline systems and **gold** is the theoretical upper-bound for the task. WSI : Word Sense Induction, WSD : Word Sense Disambiguation

System	Type	K=5	K=10	K=20	K=40	K=60
hdp-clusters-nolemma	WSI	0.508	0.6321	0.7926	0.9248	0.9821
hdp-clusters-lemma	WSI	0.4813	0.6551	0.7886	0.9168	0.9856
task11-ukp-wsi-wacky-llr	WSI	0.4119	0.5541	0.6861	0.839	0.9691
task11-ukp-wsi-wp-llr2	WSI	0.4107	0.5376	0.6887	0.8587	0.983
task11-ukp-wsi-wp-pmi	WSI	0.4045	0.5625	0.687	0.8492	0.978
task11-satty-approach1	WSI	0.3897	0.489	0.6272	0.8214	0.9745
task11.duluth.sys7.pk2	WSI	0.3888	0.5379	0.7038	0.8623	0.9844
task11.duluth.sys9.pk2	WSI	0.3715	0.499	0.6891	0.8365	0.9734
task11.duluth.sys1.pk2	WSI	0.3711	0.5329	0.7124	0.8848	0.9849
rakesh	WSD	0.4648	0.6236	0.7866	0.9072	0.9903

Table 2: S-recall@K for different values of K averaged over 100 queries.

System	Type	r=0.5	r=0.6	r=0.7	r=0.8	r=0.9
hdp-clusters-lemma	WSI	0.4885	0.4293	0.3519	0.2762	0.2376
hdp-clusters-nolemma	WSI	0.4818	0.4388	0.3485	0.293	0.2485
task11-ukp-wsi-wp-pmi	WSI	0.4283	0.334	0.2663	0.2292	0.2039
task11-ukp-wsi-wacky-llr	WSI	0.4247	0.3173	0.2539	0.2271	0.1849
task11-ukp-wsi-wp-llr2	WSI	0.4206	0.3204	0.2657	0.2241	0.1858
task11.duluth.sys1.pk2	WSI	0.4008	0.3131	0.2673	0.2451	0.2177
task11.duluth.sys7.pk2	WSI	0.3911	0.3042	0.2654	0.2343	0.1995
task11.duluth.sys9.pk2	WSI	0.359	0.2972	0.2526	0.2126	0.1951
task11-satty-approach1	WSI	0.3494	0.2688	0.2355	0.204	0.1736
rakesh	WSD	0.48	0.3904	0.3272	0.2792	0.2394

Table 3: S-precision@r for different values of r averaged over 100 queries.

References

Hui Lin and Jeff Bilmes. 2011. *A class of submodular functions for document summarization.* The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT), Portland, OR, June.

Hui Lin and Jeff A Bilmes. 2012. *Learning mixtures of submodular shells with application to document summarization.* arXiv preprint arXiv:1210.4871.

Jingrui He and Hanghang Tong and Qiaozhu Mei and Boleslaw Szymanski. 2012. *GenDeR: A Generic Diversified Ranking Algorithm.* Advances in Neural Information Processing Systems 25.

Antonio Di Marco and Roberto Navigli. 2013. *Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction.* Computational Linguistics, 39(4), MIT Press.

Roberto Navigli and Daniele Vannella. 2013. *SemEval-2013 Task 11: Evaluating Word Sense Induction Disambiguation within An End-User Application.* Proceedings of the 7th International Workshop on Semantic Evaluation (SemEval 2013), in conjunction with the Second Joint Conference on Lexical and Computational Semantcis (*SEM 2013), Atlanta, USA, 2013.

L. Lovász. 1983. *Submodular functions and convexity.* Mathematical programming-The state of the art,(eds. A. Bachem, M. Grotschel and B. Korte) Springer, pages 235257.

G.L. Nemhauser and L.A. Wolsey. 1978 *An analysis of approximations for maximizing submodular set functions I.* Mathematical Programming, 14(1):265294.