2012 IEEE 12th International Conference on Data Mining Workshops

How Much Supervision? Corpus-Based Lexeme Sentiment Estimation

Aleksander Wawer Dominika Rogozińska
Institute of Computer Science, Polish Academy of Science
ul. Ordona 21
01-237 Warszawa, Poland
axw@ipipan.waw.pl, dominikarogozinska@gmail.com

***Abstract*—This paper is focused on comparing corpus-based methods for estimating word sentiment. Evaluated algorithms represent varying degrees of supervision and range from regression alike approaches to more heavily supervised classifications. The main idea is to explore the opportunities arising from mining medium sized, balanced corpora – as opposed to web as a corpus paradigm. The comparisons have been carried using sentiment estimator benchmarks designed to take into account classification and regression problems as well as varying granularity of predicted sentiment scores: from simple to complex scales. Overall, the results turn out to be very promising and indicate superiority of supervised algorithms, especially for lower sentiment granularity predictions. However, unsupervised methods can be still considered as an interesting alternative in the case of the most fine-grained, regression like scenarios of sentiment estimation. In these cases heavy supervision and large number of features are less attractive than simple unsupervised methods.**

***Keywords*-lexeme sentiment estimation, optimization, supervised, unsupervised**

I. Introduction

This paper is focused on comparing two key approaches to lexeme sentiment estimation from corpus data. Contrary to some previous works discussed below, we assume that it is no longer feasible to use web as a corpus paradigm and therefore our efforts are aimed at maximizing the opportunities related to medium sized corpora, available for local processing on typical server class machines.

The first, well-known method of word sentiment estimation is an unsupervised one. It requires only two small sets of paradigm words, representative of each polar class (positive and negative). Our contributions, described below in section IV consist of an algorithm of systematic optimization of both sets and extending the formula to a linear regression.

The second approach, discussed in section V is a supervised one. It bases on the idea of training machine learning classifiers on vectors generated from word contexts. Specifically, we explore discriminative capabilities of classifiers trained on information obtained from two context (window) sizes, samples of contexts of various lengths and investigate the impact of morphosyntactic data. We employed three machine learning algorithms.

Our goal is to compute universal (as opposed to domain dependent) word sentiment scores using multi-point sentiment scales. We begin from the simplest binary task of distinguishing sentiment bearing words from neutral ones. Then we proceed by extending the problem to three classes: recognizing negative, neutral and positive words. We further extend the scale into a more fine-grained scenario of five classes. This is done by isolating highly polar words (highly positive and negative) into two separate classes. Finally, we extend the scale to nine points by taking into account disagreements between human raters and marking those as intermediate values.

Although the results reported below are mostly universal and applicable to other languages, our work deals with the Polish language. We test both methods on a Polish corpus and use a sentiment lexicon of Polish words, as discussed subsequently.

The paper is organized as follows. We begin with discussing major existing contributions in section II. In Section III we briefly describe two resources used in subsequently presented experiments: the National Corpus of Polish and the manually scored sentiment lexicon, used as the Golden Standard. Section IV introduces unsupervised methods of lexeme sentiment estimation and proposes improvements over existing approaches. Section V describes experiments using machine learning from word contexts. Finally, section VI presents the details of comparisons between both types of methods. We summarize the paper in section VII.

II. Existing Work

Existing methods of word sentiment estimation fall into two broad groups.

The first one is focused on WordNet and typically does not involve corpora. For instance, [1] propose and evaluate a method of determining word sentiment using WordNet based expansion of a seed set of words; it assumes that synonyms of positive words are mostly positive and antonyms mostly negative. To overcome arising problems the authors allow both positive and negative sentiment associations for each word. This enables some words to be simultaneously positive and negative. The results of evaluation against human scoring indicate comparable levels of agreement between human raters as human-machine. WordNet based expansion of sentiment labels has also been the subject of supervised classification using glosses and graph walks [2]. The resulting resource, SentiWordNet, follows the idea of

978-0-7695-4925-5/12 $26.00 © 2012 IEEE
DOI 10.1109/ICDMW.2012.119

simultaneous positive and negative associations of a synset but extends it with subjectivity information. Current state of the art resources of this type include [3].

The second group of works deals with estimating word sentiment using corpus statistics. The methods allow to compute not only word sentiment polarity, but also intensity [4], [5], [6]. Originally the approach has been proposed and evaluated in web as a corpus paradigm, using web search engines and number of results returned by appropriately crafted queries. Our work belongs to this group.

III. Resources

This section discusses two major resources that underlie the experiments described in this paper.

A. Golden Standard Lexicon

In order to measure performance of word sentiment estimation methods and train supervised models, one needs a lexicon scored by human evaluators. We use the list of 1208 Polish lexemes described by [7], developed originally to match computed ranges of continuous sentiment values to discrete scores usable by humans.

The words have been acquired using a set of manually defined lexical patterns, submitted to a search engine to find candidates for lexemes with high sentiment loading. Then, the downloaded corpus has been processed to find pattern continuations – lexemes following pattern matches, which are assumed to be candidates for sentiment words. The lexicon has been scored by two humans using a five point scale. To reflect discrepancies between scorings, the 5-point scale has been extended to 9 points. The scale ranges from -2.0 to +2.0 with the step value of 0.5.

In addition to the 9-point scale, in subsequent sections we recompute the scorings into following:

- **2-class**: Neutral words, marked as such by both annotators (0), are distinguished from all other words.
- **3-class**: Words are either negative (-2.0 to -0.5), neutral (0) or positive (0.5 to 2.0).
- **5-class**: Words are classified as very negative (-2.0 and -1.5), negative (-1.0 and -0.5), neutral (0), positive (0.5 and 1.0) and very positive (1.5 to 2.0).

B. Corpus

Our study has been conducted on the National Corpus of Polish (NKJP), the biggest and most advanced[1] resource of this type available in Polish. In particular, we conducted all the experiments on the balanced 300 million words subcorpus using the Poliqarp query engine[2].

The experiments on unsupervised sentiment estimation described subsequently in section IV rely on computing word co-occurrence frequencies. Early works such as for example in [5], used web as a corpus approach and the NEAR operator, originally provided by the Altavista search engine. The operator, placed between two keywords, returned documents where the keywords appeared within a set number of words of each other, in either order.

This does not seem feasible any more. There is no more support for NEAR and no equivalent operators in any major search engine. The possibility that still remains is to measure occurrences on web page level (equivalent to using an AND operator) – however, even in this case relying on search engine scores might not be a good idea. An exhaustive list of possible issues has been discussed by [8].

Hopefully, the Poliqarp query engine allows to substitute the NEAR operator with two queries. For example, to obtain corpus occurrences of keyword A (base form of the lexeme) within three tokens from keyword B (also base form), either order, one has to issue two following queries:

```
[base=A][]?[]?[]?[base=B]
[base=B][]?[]?[]?[base=A]
```

IV. Unsupervised Word Sentiment Estimation

This section discusses the unsupervised technique and introduces two extensions.

A. Problem Formulation

The unsupervised method employed in this paper extends the well-known approach formulated by Turney [4], where the strength of association between a word and each of the two polar classes (positive and negative, for instance) is calculated using the Semantic Orientation Pointwise Mutual Information (SO-PMI).

To begin with one needs to define the Pointwise Mutual Information (PMI) between two words, *w1* and *w2*, as:

$$\mathrm{PMI}(w1,w2) = \log_2\left(\frac{p(w1\,\&\,w2)}{p(w1)\ p(w2)}\right) \qquad (1)$$

where $p(w1\,\&\,w2)$ is the probability of co-occurrence of (w1) and (w2), while *p(w1)* and *p(w2)* denote probabilities of occurrences of *w1* and *w2*, respectively.

For estimating affective (sentiment) polarity and intensity of a word c, PMI is computed against two sets of paradigm positive and negative words, denoted as PW and NW. The formula is called the semantic orientation PMI (SO-PMI) calculated as:

$$\text{SO-PMI}(c) = \sum_{pw\in PW} \mathrm{PMI}(c,PW) - \sum_{nw\in NW} \mathrm{PMI}(c,NW) \qquad (2)$$

The selection of members of both word lists, PW and NW, is essential to the performance of the method. Existing techniques of paradigm word selection include manual selection as shown in [9], more recently decomposition and clustering of co-occurrence matrices [7]. In this paper we explore two different approaches by treating this issue as a

[1]Multiple types and levels of annotations, two search tools and query formats, deliberate selection of texts.

[2]http://poliqarp.sourceforge.net/

question of optimization, as described in subsection IV-C, and as the problem of regression as in IV-D.

B. Error Measure

All the experiments described further depend on measuring the performance of sentiment estimators. In order to compare any two estimators (in both the classification and the regression scenario), one needs to compute the errors that an estimator introduced while predicting sentiment over a lexicon. Error function E is defined over a set of words (lexicon) L as:

$$E = \sum_{w \in L} dist(s_c(w), s_e(w)) \tag{3}$$

For each word w, the distance function *dist* returns the (absolute) number of segments between the correct segment $s_c(w)$ and the segment $s_e(w)$ estimated automatically. In this context, *segment* corresponds to a range of SO-PMI values mapped to a given score on human rating scale described in Section III-A.

In Section VI the approach described here is modified so that the *dist* function returns the number of classes between the predicted and the correct one. The meaning of *dist* is intuitive and corresponds to the sum of absolute errors (SAE). However, instead of continuous values of residuals, E operates on discrete scores.

The value of E can be minimized by finding optimum locations for points separating each SO-PMI segment using Powell's conjugate direction method, determined the most effective for this task in [7]. Powell's algorithm is a non-gradient numerical optimization technique, applicable to a real-valued function which does not need to be differentiable [10].

C. Paradigm Sets Optimization

The idea behind the experiment described in this section is to formulate the choice of paradigm words as a problem of optimization. To put it more precisely, the goal is to choose two sets of paradigm words, subsets of L, that yield the best estimation performance by minimizing E. Intuitively, the selection should narrow to members of two subsets of words, labeled as the most negative (146 words) and the most positive (164 words) in human evaluation.

Unfortunately, the number of such combinations is approximately $6.9e^{21}$ and the problem is far beyond the scope of brute force computation. Precisely for this reason one needs to devise a solution that makes it possible to evaluate and select members of any paradigm set in a deliberate way. The algorithm proposed here is based on this idea.

The first step is to estimate error related to each member of PW and NW sets. This has to be done over a lexicon L which maps lexemes to human-assigned scores, described in Section III-A. The contribution of each paradigm word $pg \in PW \cup NW$, denoted as $contr(pg)$ to the total SO-PMI score (as in equation 2) can be computed according to the following formula:

$$contr(pg) = \pm \sum_{w \in L} log_2 \left(\frac{occs(pg\ \ NEAR\ w)}{log_2(occs(pg))} \right) \tag{4}$$

where sign depends on polarity: the contribution of $pg \in PW$ drives the SO-PMI score in equation 2 up, and vice versa; *occs* indicates the number of corpus occurrences of a paradigm word pg (denominator) and co-occurrences of pg with w – one within 20 tokens of the other, in either order (counter). In the above formula, NEAR only denotes word proximity in a manner consistent with former works conducted in the *web as corpus* paradigm, even though the queries have been issued using the Poliqarp search engine on a binary sub-corpus of the NKJP, as discussed in Section III. The error introduced by a paradigm word pg can be approximated by the following formula:

$$error(pg) = contr(pg) \sum_{w \in L} dist(s_c(w), s_e(w)) \tag{5}$$

The contribution depends also on the distance function *dist*, which magnifies or nullifies the error according to the results of comparison between human scoring and the SO-PMI value computed for a given word. The total error achieved using a pair of PW and NW sets is then distributed proportionally to the influence that each paradigm word pg has on the final SO-PMI value.

The core procedure of the proposed optimization can be described as follows:

```
1:  procedure OPTIMIZE(PWPN, L)
2:      PWPN ← randomize()
3:      repeat
4:          optsegments ← Powell(PWPN, L)
5:          totalerror ← E(PWPN, optsegments, L)
6:          for all pg ∈ PWPN do
7:              Errors[pg] ← error(pg)
8:          end for
9:          wppg ← argmax(Errors)
10:         for all c ∈ candidates do
11:             Scores[c] ← SO-PMI(PWPN − wppg, c)
12:         end for
13:         bpc ← argmax(Scores)
14:         PWPN[wppg] ← bpc
15:     until wppg = bpc                ▷ until minimum found
16: end procedure
```

The algorithm begins with generating paradigm sets $PWPN$ - a random combination (sample without replacement) of $PW \cup PN$. Subsequent steps (repeat loop) are as follows:

- compute *optsegments* (optimum mapping of human scores to SO-PMI segments using Powell's method);
- calculate *totalerror* of the current $PWPN$ paradigm combination over the lexicon L;

context number	context length	tags	#features	average accuracy for 10-fold cross validation								
				2 classes			3 classes			5 classes		
				SVM	ERT	RF	SVM	ERT	RF	SVM	ERT	RF
1000	3	no	157666	0.85	**0.82**	**0.76**	0.73	0.61	**0.57**	0.55	0.43	**0.40**
300	3	no	127049	0.86	0.81	**0.76**	0.76	**0.63**	0.56	0.54	0.43	0.39
1000	3	yes	158064	0.82	0.80	**0.76**	0.67	0.57	0.52	0.47	0.40	0.38
300	5	no	189019	0.86	0.80	**0.76**	0.76	0.61	0.52	0.60	0.43	0.38
300	7	no	242809	**0.89**	0.80	0.75	**0.80**	0.60	0.52	0.61	0.42	0.38
1000	5	no	209798	0.87	0.80	0.75	0.77	0.61	0.52	0.61	0.41	0.38
1000	7	no	258286	**0.89**	0.80	0.75	0.77	0.59	0.54	**0.65**	**0.44**	0.37
300	3	yes	127449	0.82	0.80	0.75	0.67	0.58	0.51	0.48	0.38	0.38
300	7	yes	243762	0.85	0.79	0.75	0.69	0.56	0.49	0.53	0.37	0.35
300	5	yes	189696	0.81	0.79	**0.76**	0.67	0.55	0.49	0.51	0.37	0.36
1000	5	yes	210471	0.84	0.79	0.75	0.68	0.54	0.50	0.54	0.36	0.36
1000	7	yes	259227	0.84	0.79	0.75	0.68	0.54	0.51	0.53	0.40	0.35

Table I
SUPERVISED ESTIMATION RESULTS: AVERAGE ACCURACY IN 10-FOLD CROSS VALIDATION. ERT - EXTREMELY RANDOMIZED TREES, RF - RANDOM FORESTS, SVM - SUPPORT VECTOR MACHINES

- estimate errors associated with each paradigm word *pg* in *PWPN*;
- select the worst performing *pg* (*wppg*) and replace it with the best performing candidate *bpc* (this demands evaluating all other positive or negative words on the current *PWPN* combination, with *pg* replaced by each of the candidates).

The method is a greedy one because at each step it seeks for the worst performing member of the $PW \cup PN$ set and replaces it with the best performing candidate. It turns out that the number of loops typically does not exceed 4. However, the algorithm turns out to be effective because the best combination obtained from 100 executions of the procedure outperforms the results obtained with SVD decomposition. This has been demonstrated in section VI.

D. Regressing on Paradigm Sets

The idea of extending the SO-PMI into linear regression (RSO-PMI) comes down to the following:

$$\text{RSO-PMI}(c) = \sum_{w \in PW \cup NW} \beta_w \text{PMI}(c, PW \cup NW) + \varepsilon \quad (6)$$

where β_w are regression coefficients associated with each word of the paradigm sets. Obviously, in the original formula of SO-PMI $\beta_w = 1$ for $w \in PW$ and $\beta_w = -1$ for $w \in NW$, $\varepsilon = 0$. The results computed using the RSO-PMI formula and two paradigm sets are presented and discussed in Section VI.

V. MACHINE LEARNING ON CORPUS CONTEXTS

This section describes an experiment with supervised learning. It contains a brief overview of the classification algorithms applied, a discussion of feature space, and finally, the results. The algorithms used in this section were based on [11].

A. Random Forests

The first classification algorithm applied is Random Forest – an ensemble method consisting of tree predictors. In this approach, each decision tree is built using randomly sampled vectors. The result is produced as follows: each tree outputs its classification (class label) for the given vector, then the most frequent class is returned [12].

B. Extremely Randomized Trees

The other applied method is Extremely Randomized Trees, another tree-based ensemble algorithm. The classifier consists of a forest of unpruned randomized decision trees. While growing the tree both feature and cut-point choice are strongly randomized and the whole learning sample is used. This approach helps to reduce the variance of the model. As demonstrated by [13] it often outperforms other tree-based classification methods, such as pruned CART trees or Random Forests. The algorithm is also computationally efficient.

C. Support Vector Classification

The third classification algorithm is the well-known Support Vector Machines (SVM). It has been shown to be effective at text categorization problems, often outperforming other methods in high-dimensional problems. In the two class scenario, the idea behind the training procedure is to find a hyperplane, represented by vector $\vec{w}$, that separates the training vectors using a maximum margin hyperplane. For more than two classes the model is made up of several binary classifiers. The implementation used in this experiment is based on a one-vs-rest scenario.

D. Input Features

Classification models were trained on random[3] occurrences of 1208 lexemes described in Section III-A. Word

[3]To rule out the possibility that the outcome may be influenced by the arrangement of documents in the corpus.

Classes	2	3	5	9
Global Opt	311	811	1211	2001
Global SVD	317	893	1494	2723
RSO-PMI Opt	315	957	1236	2549
RSO-PMI SVD	313	895	1191	2395
Extremely Randomized Trees	231	659	1037	1931
Support Vector Machines	139	348	630	1686

Table II
E SCORES (ERRORS) COMPUTED FOR VARIOUS ESTIMATORS AND SENTIMENT SCALES.

contexts were sampled from the balanced, 300 million segment version of the National Corpus of Polish. The input vectors represent frequencies of words (in both orthographic and base forms) and morphosyntactic tag occurrences at specific positions, expressed as percentages.

The experiments were intended to compare different designs of feature space and various types of features, as described below. They involve two sample sizes, 300 and 1000 contexts of each lexeme, to create vector representation. Furthermore, three different concordance window sizes (numbers of tokens left and right of the input word, taken into account when creating vectors) were tested: of 3, 5 and 7 tokens spanning left and right of the input word. Window sizes are narrow because the information about position introduces sparsity and seems to be promising only in the nearest neighborhood of the input word.

Due to the rich morphology of the Polish language, the largest contribution to the number of features comes from orthographic forms of words, i.e. the inflected forms, exactly as found in the text.

E. Feature Selection

Prior to training the classifiers, feature selection was applied by selecting the top performing percentile of features according to p-values of the ANOVA F test. For each setting (discussed below, such as window or sample size), four different percentile values were tried (1%, 0.5%, 0.1%, 0.05%). Consequently, Table I contains only the results of the top performing percentile for a given setting.

F. Results

Table I presents the results obtained with supervised learning. It contains average precision scores from 10-fold cross-validation. The experiments involved three different settings as indicated by the number of classes, described in section III-A.

Table I reveals that the number of sampled occurrences has little impact on the number of features and more importantly, on classification performance. Although some of the best performing classifiers have been trained on vectors obtained from 1000 occurrences, it seems that, overall, classifiers trained on 300 occurrences did not perform significantly worse. Most likely, 300 occurrences provide a sufficient overview of typical lexeme usage to make inferences about its sentiment orientation.

All of the three top performing classifiers did not use morphosyntactic tags. While on one hand this result may seem predictable because information about word sentiment can be typically inferred from the lexical and semantic dimension, on the other some of the syntactic clues of word usage could be potentially useful for our task. For instance, such information is important in word sense discrimination from context vectors. Apparently, this is not the case for discriminating word sentiment loading.

The other notable but trivial finding is that – as anticipated – the average precision dropped as the number of classes increased. The task of distinguishing the specific class becomes more difficult as the number of classes increases, however it is also clear that average accuracy becomes less appropriate for measuring performance. Therefore, the performance of selected best classifiers has been compared against less supervised approaches in a more regression-like scenario discussed in Section VI.

Differences between classification algorithms reveal one interesting and clear pattern: in each case, the Extreme Randomized Trees algorithm outperforms Random Forests classifiers by a wide margin of 5 to 6 percentage points.

VI. COMPARISON

This section contains a comparison of both main approaches to sentiment lexeme estimation discussed in this paper. It brings together unsupervised SO-PMI and RSO-PMI, using two combinations of paradigm words: *Opt* (the best combination computed using the optimization algorithm described in section IV-C) and the *SVD* (the combination obtained using SVD heuristic described by [7]). Both sets have been tested in two approaches:

- Using regressed RSO-PMI models as in section IV-D. Error values were computed in 10-fold cross-validation: sentiment (and thus, error) associated with each word in the lexicon was estimated using a model computed from the remaining 90% of cases, members of the training set.
- Using the original SO-PMI formula directly, in a straightforward setting: the performance of estimators has been computed over the whole lexicon – in this case, the same data that generated the sets. Arguably,

generating and testing the estimator on the same data is a methodological mistake when assessing predictive strength, nevertheless the computational cost of seeking optimum combinations is too high to apply in n-fold cross-validation.

Four unsupervised estimators were compared with the best performing supervised classifiers, namely Support Vector Classification and Extremely Randomized Trees. Each of them used the combination and number of features determined optimal for a given number of classes. As it was the case with RSO-PMI models, error values have been computed in 10-fold cross validation.

The benchmark of estimators involves an aggregated error measure E, defined in Equation 3. In the case of SO-PMI continuous scores, the distance function *dist* returns the number of (SO-PMI) *segments* between the correct segment $s_c(w)$ and the segment $s_e(w)$ estimated automatically. In the case of the classification scenario and supervised algorithms, is has been modified to return the number of *classes* between the correct and the predicted one. Because both SO-PMI segments and classes are mapped onto human scores, the meaning remains the same and both values, computed over the same lexicon, become comparable.

In this case, comparing both approaches should not rely on precision, which is not the most suitable measure in the case of multiple predicted classes of ordinal structure. Accuracy does not distinguish between types of errors and to illustrate the point, confusing positive with neutral words should not be equally treated as confusing positive with negative words.

The results are presented on Table II and Figure 1. Both illustrate the same data.

In each case, the maximum possible error can be computed as the number of classes times the size of the lexicon (1208).

The results indicate that supervised methods outperform unsupervised approaches by a significant margin. This conclusion is obviously an expected one and quite likely the SVM classifier should become the preferred approach. The difference between the SVM and the second best supervised method, ERT, is more surprising. The performance of the ERT algorithm is not far better than the globally optimized selection of paradigm words and the basic SO-PMI algorithm. However, one should interpret it with caution, as the results are not fully comparable.

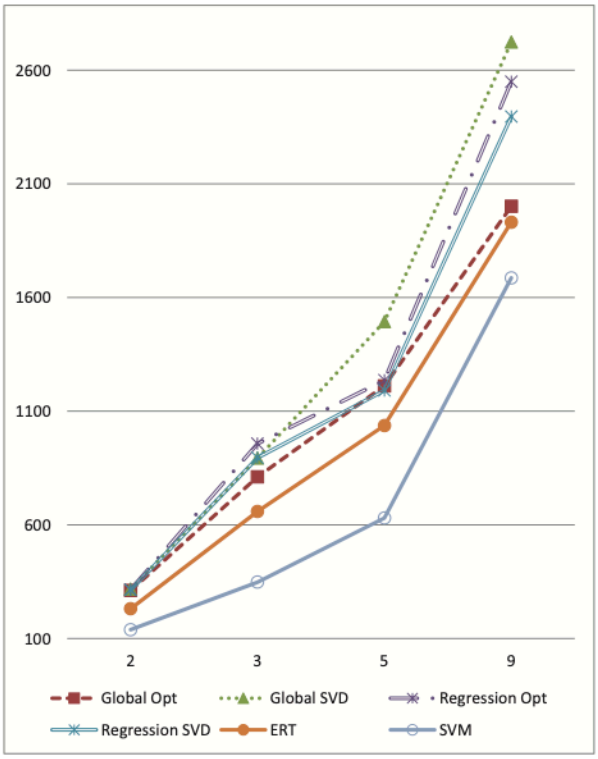

Figure 1. E scores (errors) computed for various estimators and sentiment scales.

VII. Conclusions

In this paper we extend and improve the main existing approach to unsupervised estimation of lexeme sentiment. The method, based on computing SO-PMI, has been improved in two ways. First, by proposing a randomized greedy algorithm of optimization of SO-PMI paradigm sets. Second, by extending it into a linear regression formula: RSO-PMI. Both extensions are demonstrated to bring an improvement over the existing state of the art approach, based on SVD-backed heuristic of paradigm set members for SO-PMI.

The second part of this paper describes supervised techniques, based on machine learning. We train three classification algorithms on vectors generated from random samples of word contexts from the National Corpus of Polish. We evaluate different window and sample sizes, also examine the influence of morphosyntactic information.

Finally, we evaluate all the approaches on a single benchmark. The focus is on the performance of both unsupervised and supervised techniques on sentiment estimations of various granularity. We start from the simplest binary scenario of distinguishing evaluative words (positive and negative) from neutral ones. In the next step, we increase the granularity to a multi-class setting: into 3 classes (positive, neutral and negative), then 5 and 9 classes of various sentiment intensity.

Overall, the results turn out to be very promising and indicate superiority of supervised algorithms, especially for lower sentiment granularity predictions. The best performing classification algorithm is the Support Vector Machine, which outperforms any other evaluated method. We intend to use this method to generate a high-quality sentiment lexicon.

Unsupervised (regression-like) methods turn out to be

an interesting alternative in the case of more fine-grained sentiment scales. Especially in the case of 9 point sentiment scale, the differences become less emphasized.

REFERENCES

[1] S.-M. Kim and E. Hovy, "Determining the sentiment of opinions," in *Proceedings of the 20th International Conference on Computational Linguistics*, ser. COLING '04. Stroudsburg, PA, USA: Association for Computational Linguistics, 2004.

[2] A. Esuli and F. Sebastiani, "SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining," in *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, ser. LREC'06. European Language Resources Association (ELRA), 2006.

[3] E. Cambria, C. Havasi, and A. Hussain, "SenticNet 2: A Semantic and Affective Resource for Opinion Mining and Sentiment Analysis," in *Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference*, ser. FLAIRS'12, 2012.

[4] P. Turney and M. Littman, "Measuring praise and criticism: Inference of semantic orientation from association," *ACM Transactions on Information Systems*, vol. 21, pp. 315–346, 2003.

[5] G. Grefenstette, Y. Qu, D. A. Evans, and J. G. Shanahan, *Validating the Coverage of Lexical Resources for Affect Analysis and Automatically Classifying New Words along Semantic Axes*. Springer. Netherlands, 2006, pp. 93–108.

[6] G. Wang and K. Araki, "Modifying SO-PMI for Japanese Weblog Opinion Mining by using a balancing factor and detecting neutral expressions," in *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, ser. NAACL-Short'07. Stroudsburg, PA, USA: Association for Computational Linguistics, 2007, pp. 189–192.

[7] A. Wawer, "Mining Co-Occurrence Matrices for SO-PMI Paradigm Word Candidates," in *Proceedings of the Student Research Workshop at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, ser. EACL'12 SRW. Avignon, France: Association for Computational Linguistics, April 2012, pp. 74–80.

[8] A. Kilgarriff, "Googleology is Bad Science," *Computational Linguistics*, vol. 33, no. 1, pp. 147–151, 2007. [Online]. Available: http://www.kilgarriff.co.uk/Publications/2007-K-CL-Googleology.pdf

[9] J. Read and J. Carroll, "Weakly Supervised Techniques for Domain-independent Sentiment Classification," in *Proceedings of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*. ACM, 2009, pp. 45–52.

[10] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, no. 2, pp. 155–162, Jan. 1964.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine Learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[12] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[13] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, pp. 3–42, 2006.