SteM at SemEval-2016 Task 4: Applying Active Learning to Improve Sentiment Classification

Stefan Räbiger[a], Mishal Kazmi[a], Yücel Saygın[a], Peter Schüller[b], Myra Spiliopoulou[c]
[a]Sabanci University, Istanbul, Turkey
[b]Marmara University, Istanbul, Turkey
[c]Otto-von-Guericke University, Magdeburg, Germany
{stefan,mishalkazmi,ysaygin}@sabanciuniv.edu
peter.schuller@marmara.edu.tr
myra@iti.cs.uni-magdeburg.de

Abstract

This paper describes our approach to the SemEval 2016 task 4, "Sentiment Analysis in Twitter", where we participated in subtask A. Our system relies on AlchemyAPI and SentiWordNet to create 43 features based on which we select a feature subset as final representation. Active Learning then filters out noisy tweets from the provided training set, leaving a smaller set of only 900 tweets which we use for training a Multinomial Naive Bayes classifier to predict the labels of the test set with an F1 score of 0.478.

1 Introduction

Gaining an overview of opinions on recent events or trends is an appealing feature of Twitter. For example, receiving real-time feedback from the public about a politician's speech provides insights to media for the latest polls, analysts and interested individuals including the politician herself. However, detecting tweet sentiment still poses a challenge due to the frequent use of informal language, acronyms, neologisms which constantly change, and the shortness of tweets, which are limited to 140 characters.

SemEval's subtask 4A (Nakov et al., 2016) deals with the sentiment classification of single tweets into one of the classes "positive", "neutral" or "negative". Concretely a training set of 5481 tweets and a development set comprising 1799 tweets was given, and the sentiment of 32009 tweets in a test set had to be predicted and was evaluated using F1-score. The distribution of labels for the given datasets is depicted in Table 1. Tweets with positive polarity outnumber the other two classes and the least number of instances is available for negative tweets. Initially, the organizers provided 6000 tweets for the training set and 2000 tweets for the development set, but only the tweet IDs were released to abide by the Twitter terms of agreement. By the time we downloaded the data, around 10% of the tweets (519 in the training set, 201 in the test set) were not available anymore. For further details about the labeling process and the datasets see (Nakov et al., 2016).

Lexicon-based approaches have done very well in this competition over the past years, i.e, the winners of the years 2013-2015 (Mohammad et al., 2013; Miura et al., 2014; Hagen et al., 2015) relied heavily on them. Our goal is to explore alternatives and to complement lexicon-based strategies. The SteM system performs preprocessing, including canonicalization of tweets, and based on that we extract 43 features as our representation, some features are based on AlchemyAPI[1] and SentiWordNet (Esuli and Sebastiani, 2006). We choose 28 of the features as final representation and learn three classifiers based on different subsets of these 28 features. We refer to the latter as *feature subspaces* or *subspaces* hereafter. However, independently of the subspace we use, we face the fact that tweet datasets inherently contain noise and all subspaces will - to a greater or lesser extend- be affected by this noise. To alleviate this problem, we propose to concentrate on only few of the labeled tweets, those likely to be most discriminative. To this purpose, we use Active Learning (AL) (Settles, 2012), as explained in Section 6. For AL, we set up a "budget", translating

[1]http://www.alchemyapi.com

Dataset	Total	Pos	Neu	Neg
Train	5481	2817	1882	782
Dev	1799	755	685	359

Table 1: Distribution of sentiment labels in the datasets.

Slang	c'mon	l8r	FB
Replacement	come on	later	Facebook

Table 2: Examplary slang words to be replaced by our lexicon.

to the maximum number of tweets, for which labels are requested. The term "budget" is motivated by the fact that labeling is a costly human activity. In our study, this budget is set to 900. On those 900 tweet we learn a classifier for each of the three feature subspaces to predict the labels of the test set.

The remainder of this paper is organized according to the pipeline of the SteM system. Section 2 explains preprocessing steps, Section 3 describes our features, Section 4 describes how we select our feature subset for the final representation, Section 5 gives details about learning the classifiers on the different subsets, Section 6 describes our Active Learning component, and Section 7 outlines the experiments we performed to select the best model for the competition.

2 Preprocessing

We note that while preprocessing tweets, we also extract related features. These features are described in the next section.

Removing URLs, mentions and replacing slang, abbreviations: we first remove Twitter handles (@username) and URLs. We remove dates and numbers with regular expressions and canonicalize common abbreviations, slang and negations using a lexicon we assembled from online resources.[2] Our list of negations encompasses: don't, mustn't, shouldn't, isn't, aren't, wasn't, weren't, not, couldn't, won't, can't, wouldn't. We replace these with the respective formal forms and we do the same for their apostrophe-free forms (e.g., 'cant'), which are more likely to occur in hashtags. In total we use 114 abbreviations, some are shown in Table 2.

Spelling correction: the remaining unknown words are replaced by the most likely alternative according to the PyEnchant dictionary.[3]

Splitting hashtags into words: instead of removing hashtags, we chunk them as follows. If a hashtag is comprised of a single word that exists in our dictionary, we only remove the hashtag symbol. In case of camel case hashtags (#HelloWorld), we split the words on the respective transitions from upper to lower case or vice versa. Otherwise we try to recover the multiple words in the following manner. If a hashtag contains less than 22 characters, we apply an exhaustive search to find a combination of words that all exist in a dictionary. For hashtags longer than 22 characters the exhaustive approach takes too long (about 10s), hence we opt for a greedy algorithm instead: we start at the end of the hashtag and traverse to the front trying to find the longest words that are found in a dictionary. In case not all parts can be resolved, we only keep the existing words and discard the remainder. Furthermore we remove emoticons and replace elongated characters by a single occurrence, e.g., wooooooow $\rightarrow$ wow.

Determine POS tags: on the resulting canonicalized text, we determine part-of-speech (POS) tags using the Stanford POS tagger (Manning et al., 2014). Finally we eliminate any punctuation.

Discard unbiased polarity words: since some of the words in the training corpus are uniformly distributed across the three labels, encountering such words in any tweet does not allow us to learn anything about the overall tweet sentiment. Hence, we compile a list of these words and exclude them when calculating tweet polarities. We detect such words with the help of categorical proportional difference (CPD) (Simeon and Hilderman, 2008) which describes how much a word w contributes to distinguishing the different classes. It is calculated as $CPD_w = |A - B|/(A + B)$, where A corresponds to the occurrences of the word w.r.t. one of the three classes, while B denotes the number of occurrences of the word in the remaining two classes. After computing this value for all three classes separately, the maximum value is chosen as a result. High values close to 1 indicate a strong bias towards one of the classes, while a value close to 0 signals that w is almost uniformly distributed. If this value is below

[2] http://searchcrm.techtarget.com/definition/Twitter-chat-and-text-messaging-abbreviations

[3] https://pypi.python.org/pypi/pyenchant

a fixed threshold of 0.6 we exclude the word from sentiment computation. Large CPD values indicate that a word occurs frequently with a specific class, while low values signal no particular association of the word with any class. Note that we consider only the absolute value of the enumerator in our equation, similar to (O'Keefe and Koprinska, 2009), while this is not the case in the original paper. The reason is that the direction of the association of a word with a class is not important to us.

3 Extracted features

In this section we describe the extracted features and motivate our choice. Table 3 presents an overview of the 43 extracted features. Column 'Used' lists the features that comprise our final representation after feature subset selection which is described in Section 4. We explain our reasoning for the different feature subspaces (column 'Subspace') in Section 5.

Since emoticons correlate with sentiment, we exploit this knowledge in our features. To do so, we create a lexicon encompassing 81 common positive and negative emoticons based on Wikipedia. We manually labeled these emoticons as either expressing positive or negative sentiment. While preprocessing we extract the respective emoticon features $1-2$. We assign features $3-4$ into the same category, as all four of them are easy to identify in a tweet. Hashtags share a similar relationship with sentiment like emoticons, i.e., they correlate with the overall tweet sentiment (Mohammad, 2012). Thus, we extract the features $6-16$ describing the sentiment of hashtags. Exclamation marks also hint at amplified overall tweet sentiment which is covered by feature 17. The length of a tweet affects whether it contains sentiment: if tweets are longer, it is more likely they contain mixed polarity and hence it is more difficult to label them. Features $18-20$ deal with this issue. We expect that sentiment bearing tweets contain a different sentence parts composition which is reflected in the features $21-24$.

We query AlchemyAPI about sentiment to benefit from a system that is known to yield accurate results, for example for the task of extracting entity-level sentiment (Saif et al., 2012). Moreover, AlchemyAPI allows to retrieve sentiment on different levels of granularity for documents, e.g., for a whole tweet or for single entities within a tweet. Features $25-32$ describe the relevant sentiment information from this online resource. The remaining features $33-43$ address sentiment on the whole tweet.

Note that we normalize the features $6-8, 27, 33-35$ by taking their absolute values and limiting them to the interval $[0 \dots 1]$. For extracting tweet and hashtag sentiment, we consider negations, diminishers ("a little") and intensifiers ("very") when summing up the polarity scores of the separate words. To account for the correlation of emoticons with the overall tweet sentiment, we eventually multiply the respective positive or negative overall tweet sentiment by 1.5 if emoticons are present. The resulting value is multiplied by $1.5 *$ the number of multiple exclamation marks if multiple exclamation marks exist in a sentence. We query SentiWordNet to determine word polarities in order to obtain a triple of positive, neutral and negative sentiment scores per word. This allows us to define positive/neutral/negative words according to its prevalent sentiment. Similarly, we express the overall tweet sentiment with a triple representing positive, neutral and negative polarity. Values close to 0 indicate that only little sentiment is contained in a tweet, while larger ones imply stronger sentiment. In case of negations, we employ a simple sliding windows approach to switch positive and negative sentiment of the four succeeding words. If tweets end with "not", e.g., "I like you - not", we also switch their overall sentiment as this is a common pattern in tweets indicating sarcasm. The sentiment of capitalized and elongated words is amplified. We consider a word elongated if the same letter occurs more than twice consecutively. In case of intensifiers and diminishers, the sentiment of the four succeeding words is increased in the former case by multiplying it by 1.5, and decreased in the latter one by multiplying the term by 0.5. However, if "a bit" is encountered, also the sentiment of the four preceding words is updated, e.g., "I like you a bit".

4 Selecting a feature subset

We first use Weka (Hall et al., 2009) to find an initially promising subset of features. For this purpose, we employ the WrapperSubsetEval method. That means, Multinomial Naive Bayes (MNB) is used

ID	Name	Description	Subspace	Used
1	Pos_emo	Number of positive emoticons	emoticons	yes
2	Neg_emo	Number of negative emoticons	emoticons	yes
3	Elongated	Number of elongated words	emoticons	no
4	Upper	Number of CAPITALIZED words	emoticons	no
5	has_ht	Does the tweet contain at least one hashtag?	hashtag	yes
6	neg_ht	Negative sentiment of hashtag	hashtag	no
7	neu_ht	Neutral sentiment of hashtag	hashtag	no
8	pos_ht	Positive sentiment of hashtag	hashtag	yes
9	neg_words_ht	Number of negative words in hashtag	hashtag	yes
10	neu_words_ht	Number of neutral words in hashtag	hashtag	yes
11	pos_words_ht	Number of positive words in hashtag	hashtag	yes
12	pol_words_ht	Number of polarity words in hashtag	hashtag	yes
13	negat_words_ht	Number of negation words in hashtag	hashtag	yes
14	neu_words_ht_sum	Sum of negative sentiment in hashtag	hashtag	no
15	pos_words_ht_sum	Sum of neutral sentiment in hashtag	hashtag	yes
16	pol_words_ht_sum	Sum of positive sentiment in hashtag	hashtag	yes
17	punct	Number of occurrences of '!', '??', '!?', '?!'	default	yes
18	start_len	Number of words before preprocessing	default	yes
19	end_len	Number of words after preprocessing	default	no
20	avg_len	Average number of words per sentence	default	no
21	adj_frac	Percentage of adjectives	default	yes
22	adv_frac	Percentage of adverbs	default	yes
23	v_frac	Percentage of verbs	default	no
24	nn_frac	Percentage of nouns	default	no
25	al_t_pol	Tweet polarity (AlchemyAPI)	default	no
26	al_t_type	Tweet polarity type (AlchemyAPI)	default	yes
27	al_e_pol	Average entity polarity (AlchemyAPI)	default	no
28	al_e_type	Median entity polarity type (AlchemyAPI)	default	yes
29	al_neg_e	Number of negative named entities (AlchemyAPI)	default	yes
30	al_neu_e	Number of neutral named entities (AlchemyAPI)	default	yes
31	al_pos_e	Number of positive named entities (AlchemyAPI)	default	yes
32	al_mixed	Does the tweet contain mixed sentiment? (AlchemyAPI)	default	yes
33	neg	Negative sentiment (SentiWordNet)	default	yes
34	neu	Neutral sentiment (SentiWordNet)	default	yes
35	pos	Positive sentiment (SentiWordNet)	default	yes
36	neg_words	Number of negative words (SentiWordNet)	default	yes
37	neu_words	Number of neutral words (SentiWordNet)	default	no
38	pos_words	Number of positive words (SentiWordNet)	default	yes
39	negat_words	Number of negation words (SentiWordNet)	default	no
40	pol_words	Number of polarity words (SentiWordNet)	default	yes
41	neg_words_sum	Sum of negative sentiment (SentiWordNet)	default	no
42	neu_words_sum	Sum of neutral sentiment (SentiWordNet)	default	no
43	pos_words_sum	Sum of positive sentiment (SentiWordNet)	default	yes

Table 3: Overview of our extracted features.

to compute F1-scores and with the help of 10-fold cross-validation on the training set the merit of different feature subsets is determined. This leaves us with a feature subset comprising 10 features. But we must consider the fact that we are using different feature subspaces as opposed to a single one for which Weka selected the features. Hence, our approach might benefit from other features not selected by Weka as well as some currently selected features could affect the performance of our system negatively. To investigate this, we work with SteM and perform 10-fold cross-validation on the training set to monitor effects on the F1-scores. We first try to reduce the feature subset determined by Weka further by removing features separately. This yields a feature subset encompassing 7 features. Now we add all features that Weka discarded separately back into SteM and observe the effects on the F1-scores. Following this procedure, we added 21 more features to our subset leading to the final tweet representation with 28 features. The list of used features is found in Column 'Used' in Table 3.

5 Learning a model

We employ Scikit-learn (Pedregosa et al., 2011) for building our classifiers. Since some of our features occur only in a small portion of the dataset, we build classifiers on different feature subspaces which we manually defined (column 'Subspace' in Table 3). Otherwise these underrepresented features would not be selected as informative features when removing noisy features during feature subset selection, although they actually help discriminate the different classes. For example, only 10-15% of the tweets in the training, development and test set contain hashtags. Likewise few tweets include emoticons. Hence, we consider *emoticons* and *hashtags* as separate feature subspaces. We learn in total three classifiers for three different subspaces: *default*, *default + emoticons*, *default + hashtags*. We choose MNB as our classifier as it is competitive with Logistic Regression and linear SVM and fast in learning models which allows us to carry out multiple experiments for quantifying the merit of different AL strategies, feature subsets, etc., as these experiments are time-consuming. With a similar reasoning we decide against ensemble methods for now as we first want to obtain reliable results for single classifiers before studying ensembles.

6 Active learning

AL is motivated well in (Settles, 2010): "The key idea behind *active learning* is that a machine learning algorithm can achieve greater accuracy with fewer training labels if it is allowed to choose the data from which it learns." In (Martineau et al., 2014), the authors apply AL to detect misclassified samples and let experts relabel those instances to reduce noise. We utilize AL in a similar fashion, but instead of relabeling tweets, we discard them. We set a fixed budget for the AL strategy, which indicates for how many tweets the classifer can use the label for training, after starting from a small seed set of labeled tweets.

As AL strategies we pick uncertainty sampling (UC) and certainty sampling (C) and choose MNB as classifier. We calculate certainty/uncertainty according to two different criteria, namely margin and confidence (Li et al., 2012). In margin-based UC the tweet with the highest margin is selected for labeling, while in confidence-based UC the instance with the least confidence is chosen. Contrary to UC, C always selects the tweet about which the classifier is most confident in case of confidence, or the tweet with the lowest margin respectively. We initialize the seed set with approximately the same number of tweets from all three classes where the tweets are chosen randomly per class. Whenever an AL strategy selects a tweet to be labeled, we reveal its actual label. To identify a fixed budget for our AL strategies, we test different configurations of budgets and seed set sizes on the training set. For each run we perform 10-fold cross validation and average results over three executions to account for chance and then the labels of the development set are predicted. As a baseline method we choose random sampling which selects arbitrary tweets to be labeled. After conducting multiple experiments, we find that initializing the seed set with 500 tweets, setting the budget to 400 tweets and choosing confidence-based UC yields the highest weighted F1-scores with $F1 = 0.48$. Our experimental evaluation of different AL methods is visualized in Figure 1 using a seed set with 500 tweets

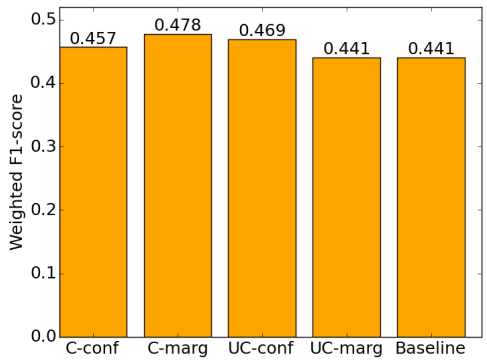

Figure 1: F1-scores on development set for different AL strategies with a seed set size of 500 tweets and a budget of 400 tweets.

Strategy	#Features	#Tweets	Dev
BL	8	5481	0.444
BL	8	900	0.466
SteM	43	5481	0.454
SteM	28	900	**0.473**

Table 4: Comparing F1-scores on development set using SteM and our baseline.

		Predicted		
		neg	neu	pos
Truth	neg	181	93	85
	neu	206	181	298
	pos	116	124	515

Table 5: Confusion matrix of SteM with 900 training instances.

and a budget of 400 tweets for which the strategies request the revealed labels. Although margin-based C seems to outperform confidence-based UC, we select the latter strategy. Tests on the development set using only the tweets selected by the respective AL strategies revealed that C achieved an F1-score of 0.30 while confidence-based UC achieved around 0.48. This inferiority of C on our data confirms reports from the literature, see e.g., (Kumar et al., 2010; Ambati et al., 2011).

7 Experiments

In this section we evaluate our approach on the development set as no labels for the test set are available. As AlchemyAPI is a full-fledged system, we use the 8 features extracted from it (Features $25-32$ in Table 3) as a baseline in our experiments to compare it with SteM using a) the full training set and b) the reduced tweets after performing confidence-based UC as explained in the previous section. We then reapply the learning procedure described in Section 5 to obtain our F1-scores. The results are depicted in Table 4. Initially, our system achieves an F1-score of 0.454 using all training instances. After selecting the 900 most informative tweets using confidence-based UC from the previous section, the score increases to 0.473. We observe a similar trend for our baseline and note that it is outperformed by SteM, although the margin shrinks when reducing the number of tweets in the training set. When analyzing the corresponding confusion matrix of SteM with 900 tweets in Table 5, it becomes obvious that it fails to distinguish neutral sentiment from the remaining ones properly.

8 Conclusion and future work

In this paper we proposed SteM to predict tweet sentiment. After preprocessing, it extracts 43 features from tweets, selects 28 of these features as an appropriate subset to represent tweets for Multinomial Naive Bayes. One such classifier is trained for each of our three overlapping feature subspaces. To predict the labels of unknown instances, they are passed to the classifier that was trained on the respective feature subspace. Due to the noisy nature of labels in sentiment analysis, we select only few tweets for our training set by applying Active Learning. Despite utilizing only 26.3% (900 out of 5481) of the provided tweets for training, SteM outperforms an identical system trained on the full training set. Overall, our approach looks promising, but has room for improvement. Firstly, we plan to test our approach with ensembles and secondly, identify tweets with neutral sentiment more accurately. To this purpose, we plan to incorporate more features.

Acknowledgments

This work is partially supported by TUBITAK Grant 114E777.

References

Vamshi Ambati, Sanjika Hewavitharana, Stephan Vogel, and Jaime Carbonell. 2011. Active learning with multiple annotations for comparable data classification task. In *Proceedings of the 4th Workshop on Building and Using Comparable Corpora: Comparable Corpora and the Web*, pages 69–77. Association for Computational Linguistics.

Andrea Esuli and Fabrizio Sebastiani. 2006. Sentiwordnet: A publicly available lexical resource for opinion mining. In *In Proceedings of the 5th Conference on Language Resources and Evaluation (LREC06*, pages 417–422.

Matthias Hagen, Martin Potthast, Michael Büchner, and Benno Stein. 2015. Webis: An ensemble for twitter sentiment detection. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, pages 582–589.

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

Mohit Kumar, Rayid Ghani, Mohak Shah, Jaime G Carbonell, and Alexander I Rudnicky. 2010. Empirical comparison of active learning strategies for handling temporal drift. *ACM Transactions on Embedded Computing Systems*, 9(4):161–168.

Shoushan Li, Shengfeng Ju, Guodong Zhou, and Xiaojun Li. 2012. Active learning for imbalanced sentiment classification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 139–148. Association for Computational Linguistics.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60.

Justin Martineau, Lu Chen, Doreen Cheng, and Amit Sheth. 2014. Active learning with efficient feature weighting methods for improving data quality and classification accuracy. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1104–1112, Baltimore, Maryland, June. Association for Computational Linguistics.

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. Teamx: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 628–632, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*, Atlanta, Georgia, USA, June.

Saif M Mohammad. 2012. # emotional tweets. 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 246–255. Association for Computational Linguistics.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*. Association for Computational Linguistics.

Tim O'Keefe and Irena Koprinska. 2009. Feature selection and weighting methods in sentiment analysis. In *Proceedings of the 14th Australasian document computing symposium, Sydney*, pages 67–74. Citeseer.

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. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Hassan Saif, Yulan He, and Harith Alani. 2012. Semantic sentiment analysis of twitter. In *The Semantic Web–ISWC 2012*, pages 508–524. Springer.

Burr Settles. 2010. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11.

Burr Settles. 2012. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114.

Mondelle Simeon and Robert Hilderman. 2008. Categorical proportional difference: A feature selection method for text categorization. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 201–208. Australian Computer Society, Inc.