Sparse Non-negative Matrix Language Modeling: Maximum Entropy Flexibility on the Cheap

Ciprian Chelba[1]*, Diamantino Caseiro*[2]*, Fadi Biadsy*[2]

[1]Google, Inc., Mountain View
[2]Google, Inc., New York City

ciprianchelba,dcaseiro,biadsy@google.com

Abstract

We present a new method for estimating the sparse non-negative model (SNM) by using a small amount of held-out data and the multinomial loss that is natural for language modeling; we validate it experimentally against the previous estimation method which uses leave-one-out on training data and a binary loss function and show that it performs equally well. Being able to train on held-out data is very important in practical situations where training data is mismatched from held-out/test data. We find that fairly small amounts of held-out data (on the order of 30-70 thousand words) are sufficient for training the adjustment model, which is the only model component estimated using gradient descent; the bulk of model parameters are relative frequencies counted on training data.

A second contribution is a comparison between SNM and the related class of Maximum Entropy language models. While much cheaper computationally, we show that SNM achieves slightly better perplexity results for the same feature set and same speech recognition accuracy on voice search and short message dictation.

Index Terms: language modeling, maximum entropy, speech recognition, machine learning

1. Introduction

A statistical language model estimates probability values $P(W)$ for strings of words W in a vocabulary $\mathcal{V}$ whose size is in the tens, hundreds of thousands and sometimes even millions. Typically the string W is broken into sentences, or other segments such as utterances in automatic speech recognition, which are often assumed to be conditionally independent; we will assume that W is such a segment, or sentence.

Since the parameter space of $P(w_k|w_1, w_2, \ldots, w_{k-1})$ is too large, the language model (LM) is forced to put the *context* $W_{k-1} = w_1, w_2, \ldots, w_{k-1}$ into an *equivalence class* determined by a function $\Phi(W_{k-1})$. As a result,

$$P(W) \cong \prod_{k=1}^{n} P(w_k|\Phi(W_{k-1})) \quad (1)$$

Research in language modeling consists of finding appropriate equivalence classifiers Φ and methods to estimate $P(w_k|\Phi(W_{k-1}))$. Once the form $\Phi(W_{k-1})$ is specified, only the problem of estimating $P(w_k|\Phi(W_{k-1}))$ from training data remains.

The contribution of this paper is two-fold:

- Sections 2 and 3 present a new way of estimating the sparse non-negative model (SNM) by using a small amount of held-out data and the multinomial loss that is natural for LM; we validate it against the previous estimation method in [1]-[2]
- Section 4 contrasts the SNM LM with the related class of maximum entropy (MaxEnt) LMs; experiments on the One Billion Words corpus [3] show that SNM achieves slightly better perplexity results for the same feature set, and same speech recognition accuracy on voice search and short message dictation.

2. Notation and Modeling Assumptions

We denote with e an event in the training/development/test data corresponding to each prediction $(w_k|\Phi(W_{k-1}))$ in Eq. (1); each event consists of:

- a set of *features* $\mathcal{F}(e) = \{f_1, \ldots, f_k, \ldots, f_{F(e)}\} \subset \mathcal{F}$, where $\mathcal{F}$ denotes the set of features in the model, collected on the training data: $\mathcal{F} = \cup_{e \in \mathcal{T}} \mathcal{F}(e)$;
- a *predicted (target) word* $w = t(e)$ from the LM vocabulary $\mathcal{V}$; we denote with $V = |\mathcal{V}|$ the size of the vocabulary.

The set of features $\mathcal{F}(e)$ is obtained by applying the equivalence classification function $\Phi(W_{k-1})$ to the context of the prediction. The n-gram model is a particular case extracting all n-gram features of length $0, \ldots, n-1$ from the W_{k-1} context[1], respectively.

2.1. Skip-n-gram Language Modeling

A simple variant on the n-gram model is the skip-n-gram model; a skip-n-gram feature extracted from the context W_{k-1} is characterized by the tuple (r, s, a) where:

- r denotes number of remote context words
- s denotes the number of skipped words
- a denotes the number of adjacent context words

relative to the target word w_k being predicted. For example, in the sentence,
`<S> The quick brown fox jumps over the lazy dog </S>` a $(1, 2, 3)$ skip-gram feature for the target word `dog` is:
`[brown skip-2 over the lazy]`

To control the size of $\mathcal{F}(e)$ it is recommended to limit the skip length s and also either $(r + a)$ or both r and s. We configure the skip-n-gram feature extractor to produce all features $\mathbf{f}$ defined by the equivalence class $\Phi(W_{k-1})$ that meet constraints on minimum and maximum values for:

- the number of context words used $r + a$;
- the number of remote words r;

[1]The empty feature is considered to have length 0, it is present in every event e, and it produces the unigram distribution on the language model vocabulary.

- the number of adjacent words a;
- the skip length s.

We also allow the option of not including the exact value of s in the feature representation; this may help with smoothing by sharing counts for various skip features. Tied skip-n-gram features look like: `[curiosity skip-* the cat]`

A simple extension that leverages context beyond the current sentence, as well as other categorical features such as geolocation is presented and evaluated in [4].

In order to build a good probability estimate for the target word w_k in a context W_{k-1}, or an event e in our notation, we need a way of combining arbitrary features which do not fall into a simple hierarchy like regular n-gram features. The following section describes a simple yet novel approach for combining such predictors in a way that is computationally easy, scales up gracefully to large amounts of data and as it turns out is also very effective from a modeling point of view.

3. Multinomial Loss for the Sparse Non-negative Matrix Language Model

The sparse non-negative matrix (SNM) language model (LM) [1]-[2] assigns probability to a word by applying the equivalence classification function $\Phi(W)$ to the context of the prediction, as explained in the previous section, and then using a matrix $\mathbf{M}$, where M_{fw} is indexed by feature $f \in \mathcal{F}$ and word $w \in \mathcal{V}$. We further assume that the model is parameterized as a slight variation on conditional relative frequencies for words w given features f, denoted as $c(w|f)$:

$$P(w|\Phi(W)) \propto \sum_{f \in \Phi(W)} \underbrace{c(w|f) \cdot exp(A(f,w;\boldsymbol{\theta}))}_{M_{fw}} \quad (2)$$

The *adjustment function* $A(f,w;\boldsymbol{\theta})$ is a real-valued function whose task is to estimate the relative importance of each input *feature* f for the prediction of the given *target word* w. It is computed by a linear model on *meta-features* h extracted from each *link* (f,w):

$$A(f,w;\boldsymbol{\theta}) = \sum_k \theta_k h_k(f,w) \quad (3)$$

The meta-features are either strings identifying the feature type, feature, link etc., or bucketed feature and link counts. We also allow all possible conjunctions of elementary meta-features, and estimate a weight θ_k for each (elementary or conjoined) meta-feature h_k. In order to control the model size we use the hashing technique in [5],[6]. The meta-feature extraction is explained in more detail in [7].

Assuming we have a sparse matrix $\mathbf{M}$ of adjusted relative frequencies, the probability of an event $e = (w|\Phi(W))$ predicting word w in context $\Phi(W)$ is computed as follows:

$$\begin{aligned} P(e) &= y_t(e)/y(e) \\ y_t(e) &= \sum_{f \in \mathcal{F}} \sum_{w \in \mathcal{V}} 1_f(e) \cdot 1_w(e) M_{fw} \\ y(e) &= \sum_{f \in \mathcal{F}} 1_f(e) M_{f*} \end{aligned}$$

where M_{f*} ensures that the model is properly normalized over the LM vocabulary:

$$M_{f*} = \sum_{w \in \mathcal{V}} M_{fw}$$

and the indicator functions $1_f(e)$ and $1_w(e)$ select a given feature and target word in the event e, respectively.

With this notation and using the shorthand $A_{fw} = A(f,w;\boldsymbol{\theta})$, the derivative of the log-probability for event e with respect to the adjustment function A_{fw} for a given link (f,w) is:

$$\begin{aligned} \frac{\partial \log P(e)}{\partial A_{fw}} &= \frac{\partial \log y_t(e)}{\partial A_{fw}} - \frac{\partial \log y(e)}{\partial A_{fw}} \\ &= \frac{1}{y_t(e)} \frac{\partial y_t(e)}{\partial A_{fw}} - \frac{1}{y(e)} \frac{\partial y(e)}{\partial A_{fw}} \\ &= 1_f(e) M_{fw} \left[\frac{1_w(e)}{y_t(e)} - \frac{1}{y(e)} \right] \end{aligned} \quad (4)$$

making use of the fact that $\frac{\partial M_{fw}}{\partial A_{fw}} = \frac{\partial c(w|f) exp(A_{fw})}{\partial A_{fw}} = c(w|f) exp(A_{fw}) = M_{fw}$

Propagating the gradient $\frac{\partial \log P(e)}{\partial A_{fw}}$ to the $\boldsymbol{\theta}$ parameters of the adjustment function $A(f,w;\boldsymbol{\theta})$ is done using mini-batch estimation for efficiency reasons:

$$\begin{aligned} \frac{\partial \log P(e)}{\partial \theta_k} &= \sum_{(f,w):h_k \in meta-feat(f,w)} \frac{\partial \log P(e)}{\partial A_{fw}} \\ \theta_{k,B+1} &\leftarrow \theta_{k,B} - \eta \sum_{e \in B} \frac{\partial \log P(e)}{\partial \theta_k} \end{aligned} \quad (5)$$

Rather than using a single fixed learning rate η, we use AdaGrad [8] which uses a separate adaptive learning rate $\eta_{k,B}$ for each weight $\theta_{k,B}$:

$$\eta_{k,B} = \frac{\gamma}{\sqrt{\Delta_0 + \sum_{b=1}^{B} \left[\sum_{e \in b} \frac{\partial \log P(e)}{\partial \theta_k} \right]^2}} \quad (6)$$

where B is the current batch index, γ is a constant scaling factor for all learning rates and Δ_0 is an initial accumulator constant. Basing the learning rate on historical information tempers the effect of frequently occurring features which keeps the weights small and as such acts as a form of regularization.

We refer the reader to Sections 3.1-2 of [7] for further details on the implementation and a description of the meta-features used by the adjustment model.

4. Maximum Entropy Language Model

Maximum entropy (MaxEnt) is a well studied modeling technique that similarly to SNM allows the use of arbitrary features for estimating $P(w|\Phi(W))$. It differs from the SNM parameterization of $P(w|\Phi(W))$ for an event $e = (w|\Phi(W))$, see Eq. 2 for a direct comparison:

$$P(w|\Phi(W)) = \frac{exp[\sum_{f,w} 1_w(e) \cdot 1_f(e) \cdot \lambda_{f,w}]}{Z_{\Phi(W)}} \quad (7)$$

where $\lambda_{f,w}$ is the parameter associated with the joint MaxEnt feature (f,w) and $Z_{\Phi(W)}$ is a normalization factor or partition function.

We train our MaxEnt models using stochastic gradient descent. The computational complexity of each update is $O(|\mathcal{V}|)$, similar to multinomial SNM updates. In practice one SNM update on a given held-out event is more expensive due to the large number of meta-feature being updated; on the other hand MaxEnt training requires multiple iterations though the full training data set, making it orders of magnitude more expensive to train.

Model	Lexicalized Meta-features	Test Set PPL
Interpolated Kneser-Ney, baseline		
5-gram		67.6
SNM, Leave-one-out on Training Data		
5-gram	yes	70.8
skip-5-gram	yes	52.9
SNM, Multinomial Loss on Held-out Data		
5-gram	*Unadjusted Model*	86.0
	yes (one epoch)	71.4
	no	**69.6**
skip-5-gram	*Unadjusted Model*	69.2
	yes (one epoch)	54.4
	no	**50.9**

Table 1: *Experiments on the One Billion Words Language Modeling Benchmark in* `5-gram` *and* `skip-10-gram` *configuration; 20 million maximum number of hashed parameters, 2048 mini-batch size, one or five training epochs.*

We distribute the training of MaxEnt model using hundreds of machines using the Iterative Parameter Mixture (IPM) method [9] as described in [10].

4.1. Hierarchical Modeling

Even using the IPM method, MaxEnt model training is still too expensive for very large corpora. We partition the vocabulary in clusters $c(w)$ and use a hierarchical model:

$$P(w|\Phi(W)) \propto P(c(w)|\Phi(W)) \cdot P(w|\Phi(W), c(w)) \quad (8)$$

where $P(c(w)|\Phi(W))$ is a model over the cluster vocabulary and $P(w|\Phi(W), c(w))$ is a model similar to Equation 7, but normalized over the words in cluster $c(w)$. Now, instead of one update with complexity $O(|\mathcal{V}|)$, training requires two updates with complexity as low as $O(|\sqrt{(}\mathcal{V})|$. Hierarchical modeling can improve model quality [11]. The quality of the hierarchical model depends crucially on the clustering method. In this paper we use the algorithm described in [12]. SNM can also benefit from hierarchical modeling in which case $P(c(w)|\Phi(W))$ and $P(w|\Phi(W), c(w))$ are independent SNM models.

4.2. Contrasting SNM versus MaxEnt Model Estimation

SNM and MaxEnt differ in the estimation procedure. Most SNM parameters are stored in the matrix **M** which is initialized with counts from the training data. These parameters are fixed during the adjustment procedure, which only needs to estimate a relatively small set of parameters whose size is controlled by the types of meta-features used in the model. This can be done reliably and efficiently with a small held-out set of high quality in-domain data. MaxEnt does not have a known comparable initialization procedure; training requires updating all parameters from scratch by iterating through the training data until each feature is seen enough times; it is often beneficial to have a second training stage that adapts the model to in-domain transcribed data.

5. Experiments

5.1. Experiments on the One Billion Words Language Modeling Benchmark

Our first experimental setup used the One Billion Word Benchmark (OBWB) corpus[2] made available by [3]. For completeness, here is a short description of the corpus, containing only monolingual English data:

- total number of training tokens is about 0.8 billion
- the vocabulary provided consists of 793471 words including sentence boundary markers `<S>`, `</S>`, and was constructed by discarding all words with count below 3
- words outside of the vocabulary were mapped to an `<UNK>` token, also part of the vocabulary
- sentence order was randomized
- the test data consisted of 159658 words (without counting the sentence beginning marker `<S>` which is never predicted by the language model)
- the out-of-vocabulary (OOV) rate on the test set was 0.28%.

The foremost concern when using held-out data for estimating the adjustment model is the limited amount of data available in a practical setup, so we used a small development set consisting of 33 thousand words.

We conducted experiments using two feature extraction configurations identical to those used in [1]: `5-gram` and `skip-10-gram`, see Appendix in [7]. The AdaGrad parameters in Eq. (6) are set to: $\gamma = 0.1$, $\Delta_0 = 1.0$, and the mini-batch size is 2048 samples. We also experimented with various adjustment model sizes (200M, 20M, and 200k hashed parameters) and non-lexicalized meta-features. The results are presented in Table 1; since there was little dependence on the adjustment model size, we only list results for the 20M hashed parameters setting.

A first conclusion is that training the adjustment model on held-out data using multinomial loss matches the previous results reported in [1].

A second conclusion is that we can indeed get away with very small amounts of development data. This is excellent news, because development data is typically in short supply.

Model	Number Parameters (billions)	Test Set PPL
SNM 5-gram	2.6	67.4
MaxEnt 5-gram	2.1	77.1

Table 2: *Experiments on the One Billion Words Language Modeling Benchmark with hierarchical configurations.*

5.2. Comparison with Equivalent Maximum Entropy Model

Table 2 shows the perplexity of various hierarchical models on the OBWB, all experiments use the same 1000 cluster vocabulary partition. We observe that hierarchical modeling slightly improves SNM modeling relative to the best 5-gram model in Table 1. We also see that the SNM model has significantly lower perplexity than the MaxEnt model. The number of parameters is higher for SNM because it stores an extra parameter per context.

5.3. ASR Experiments

We conducted automatic speech recognition (ASR) experiments to compare SNM and MaxEnt hierarchical models. All experiments were based on Google's cloud based mobile ASR system for Italian. This is a state of the art system with an LSTM acoustic model and a 15 million n-gram LM for the first pass, estimated from various data sources using Bayesian interpolated [13]. In this work the proposed second pass SNM and

[2] http://www.statmt.org/lm-benchmark

Stage	MaxEnt	SNM
Word Clustering	4:00	4:00
Initial Model Building	0.40	1:50
Training	4:50 (500 workers)	
Adapt	0:30 (50 workers)	
Adjust		0:40 (1 worker)
Total	12:00	6:30

Table 3: *Model training times (h:mm) along with number of machines used for each training stage.*

MaxEnt models, respectively, are used to rescore n-best lists generated by the first pass system. The score of the second pass LM is log linearly interpolated with the first pass LM score and with the acoustic model score.

The corpus used to train the second pass models consists of 35 billion words of mobile written sentences and a small subset of 15 million transcribed words. All data were anonymized and stripped of personally identifying items.

The vocabulary contains 3.9 million words. We rank the vocabulary according to the distribution in automatically recognized ASR logs. The most frequent million words are clustered in 1000 clusters. The remaining words are assigned to a special cluster <TAIL>. For efficiency its cluster conditional sub-model $P(w|\Phi(W), c(w) = \texttt{<TAIL>})$ is estimated using unigram relative frequencies instead of an SNM or MaxEnt model.

The types of features used fall into the following feature templates:

- word n-grams, $y, w_{i-1}, \cdots, w_{i-k}$ up to 5-gram.
- word cluster n-grams, $y, c(w_{i-1}), \cdots, c(w_{i-k})$ from 3 to 5-gram.
- skip 2-grams, $y, *, w_{i-k}$ up to 5 word gap.
- left and right skip 3-grams, $y, *, w_{i-k+1}, w_{i-k}$, and $y, w_{i-1}, *, w_{i-k}$ up to 3 word gap.

y is the token being predicted $c(w_i)$ in the case of the cluster sub-model and w_i in the case of the cluster conditioned sub-models.

We trained MaxEnt models with and without *PrefixBack-off*$_0$ features as described in [10]. These features are shared between contexts W_{k-1} in the same feature template and trigger when a regular feature is missing. We selected approximately 5 billion features per model using counts cut offs. The MaxEnt models were first trained for 5 epochs on the training data and then adapted by continuing training on the transcribed data subset for 3 additional epochs using a small learning rate.

Counts for the unadjusted SNM model were collected during initialization. The model was then adjusted using a 500k word held-out subset of the transcribed data. We used 20 million hashed parameters and trained for 3 epochs using AdaGrad. We observed no improvements by increasing the size of the held-out set or the number of epochs.

We shared as many components as possible between the MaxEnt and SNM implementations, including: all data processing, clustering, feature extraction, ASR rescorer, etc. The differences between the two systems are mostly limited to the probability and gradient computation functions.

Table 3 shows a breakdown of the training time for the MaxEnt and SNM LMs, respectively. Our MaxEnt training pipeline is highly optimized but it is clear that multinomial SNM training requires a very small fraction of the computational resources of MaxEnt: MaxEnt training and adapting requires about 2,442 machine hours whereas SNM adjustment requires only 2/3 machine hours. While still very significant, the end to end training time reduction is not as impressive due to MaxEnt's distributed implementation and to lengthy sibling stages such as word clustering.

Model	WER (%)	
	Voice Search	Short Message
SNM	8.0	14.9
MaxEnt	8.0	14.9
Back-off MaxEnt	7.9	14.8

Table 4: *Word Error Rate of MaxEnt and SNM hierarchical models.*

Table 4 presents the word error rate (WER) of various models measured on voice search and short message held-out test sets. We observe that SNM matches the accuracy of standard MaxEnt. When augmented with back-off features MaxEnt improves slightly; experiments adding similar back-off features to SNM were unproductive.

6. Conclusions and Future Work

The main conclusion is that training the adjustment model on held-out data using multinomial loss introduces many advantages while matching the previous results reported in [1]: as observed in [14], Section 2, using a binary probability model is expected to yield the same model as a multinomial probability model. Correcting the deficiency in [1] induced by using a Poisson model for each binary random variable does not seem to make a difference in the quality of the estimated model.

Being able to train on held-out data is very important in practical situations where the training data is mismatched from the held-out/test data. It is also less constrained than the previous training algorithm using leave-one-out on training data: it allows the use of richer meta-features in the adjustment model, e.g. the diversity counts used by Kneser-Ney smoothing which would be difficult to deal with correctly in leave-one-out training, or taking into account the data source for a given skip-/n-gram feature and combining them for best performance on held-out/test data (similar to the Bayesian interpolation algorithm [13] used for estimating the first-pass model in our ASR system).

We find that fairly small amounts of held-out data (on the order of 30-70 thousand words) are sufficient for training the adjustment model.

Just like MaxEnt, SNM can be extended to hierarchical modeling and we observe significant perplexity improvements: multinomial adjusted SNM achieves 14% relative lower perplexity than similar MaxEnt models on the One Billion Words Benchmark. In terms of ASR quality SNM matches the WER of standard MaxEnt while being orders of magnitude cheaper to train. However, MaxEnt augmented with back-off features is slightly better. In future work we plan to research techniques for adding similar features to SNM models.

7. Acknowledgments

Thanks go to Yoram Singer for clarifying the correct mini-batch variant of AdaGrad, Noam Shazeer for assistance on understanding his implementation of the adjustment function estimation, Kunal Talwar, Amir Globerson for useful discussions and former summer intern Joris Pelemans for suggestions while preparing the paper.

8. References

[1] N. Shazeer, J. Pelemans, and C. Chelba, "Skip-gram language modeling using sparse non-negative matrix probability estimation," *CoRR*, vol. abs/1412.1454, 2014. [Online]. Available: http://arxiv.org/abs/1412.1454

[2] N. M. Shazeer, J. Pelemans, and C. Chelba, "Sparse non-negative matrix language modeling for skip-grams," in *Proceedings of Interspeech 2015*, 2015, pp. 1428–1432.

[3] C. Chelba, T. Mikolov, M. Schuster, Q. Ge, T. Brants, and P. Koehn, "One billion word benchmark for measuring progress in statistical language modeling," *CoRR*, vol. abs/1312.3005, 2013. [Online]. Available: http://arxiv.org/abs/1312.3005

[4] C. Chelba and N. Shazeer, "Sparse non-negative matrix language modeling for geo-annotated query session data," in *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Dec 2015, pp. 8–14.

[5] K. Ganchev and M. Dredze, "Small statistical models by random feature mixing," in *Proceedings of the ACL-2008 Workshop on Mobile Language Processing, Association for Computational Linguistics*, 2008, pp. 604–613.

[6] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in *Proceedings of the 26th Annual International Conference on Machine Learning*, ser. ICML '09. New York, NY, USA: ACM, 2009, pp. 1113–1120. [Online]. Available: http://doi.acm.org/10.1145/1553374.1553516

[7] C. Chelba and F. Pereira, "Multinomial loss on held-out data for the sparse non-negative matrix language model," *CoRR*, vol. abs/1511.01574, 2015. [Online]. Available: http://arxiv.org/abs/1511.01574

[8] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, Jul. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2021068

[9] K. Hall, S. Gilpin, and G. Mann, "Mapreduce/bigtable for distributed optimization," in *Neural Information Processing Systems Workshop on Leaning on Cores, Clusters, and Clouds*, 2010.

[10] F. Biadsy, K. Hall, P. Moreno, and B. Roark, "Backoff inspired features for maximum entropy language models," in *Proceedings of Interspeech*, 2014.

[11] S. F. Chen, "Shrinking exponential language models," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 468–476.

[12] J. Uszkoreit and T. Brants, "Distributed word clustering for large scale class-based language modeling in machine translation," in *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, 2008, pp. 755–762.

[13] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," in *Interspeech 2011*, 2011, pp. 1429–1432.

[14] P. Xu, A. Gunawardana, and S. Khudanpur, "Efficient subsampling for training complex language models," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1128–1136. [Online]. Available: http://dl.acm.org/citation.cfm?id=2145432.2145552