Breaking Sticks and Ambiguities with Adaptive Skip-gram

Sergey Bartunov[†]
National Research University
Higher School of Economics[†]
Moscow, Russia

Dmitry Kondrashkin
Yandex
Moscow, Russia

Anton Osokin
INRIA – Sierra Project-Team,
École Normale Supérieure
Paris, France

Dmitry P. Vetrov[†]
Skolkovo Institute of
Science and Technology
Moscow, Russia

Abstract

The recently proposed Skip-gram model is a powerful method for learning high-dimensional word representations that capture rich semantic relationships between words. However, Skip-gram as well as most prior work on learning word representations does not take into account word ambiguity and maintain only a single representation per word. Although a number of Skip-gram modifications were proposed to overcome this limitation and learn multi-prototype word representations, they either require a known number of word meanings or learn them using greedy heuristic approaches. In this paper we propose the Adaptive Skip-gram model which is a non-parametric Bayesian extension of Skip-gram capable to automatically learn the required number of representations for all words at desired semantic resolution. We derive efficient online variational learning algorithm for the model and empirically demonstrate its efficiency on word-sense induction task.

1 Introduction

Continuous-valued word representations are very useful in many natural language processing applications. They could serve as input features for higher-level algorithms in text processing pipeline and help to overcome the word sparseness of natural texts. Moreover, they can explain on their own many semantic properties and relationships between concepts represented by words.

Recently, with the success of the deep learning, new methods for learning word representations inspired by various neural architectures were introduced. Among many others the two particular models Continuous Bag of Words (CBOW) and Skip-gram (SG) proposed in (Mikolov et al., 2013a) were used to obtain high-dimensional distributed representations that capture many semantic relationships and linguistic regularities (Mikolov et al., 2013a,b). In addition to high quality of learned representations these models are computationally very efficient and allow to process text data in online streaming setting.

However, word *ambiguity* (which may appear as polysemy, homonymy, etc) an important property of a natural language is usually ignored in representation learning methods. For example, word "apple" may refer to a fruit or to the Apple inc. depending on the context. Both CBOW and SG also fail to address this issue since they assume a unique representation for each word. As a consequence either the most frequent meaning of the word dominates the others or the meanings are mixed. Clearly both situations are not desirable for practical applications.

We address the problem of unsupervised learning of multiple representations that correspond to different meanings of a word, i.e. building multi-prototype word representations. This may be considered as specific case of word sense induction (WSI) problem which consists in automatic identification of the meanings of a word. In our case different meanings are distinguished by separate representations. We define *meaning* or *sense* as distinguishable interpretation of the spelled word which may be caused by any kind of ambiguity.

Word-sense induction is closely related to the word-sense disambiguation (WSD) task where the goal is to choose which meaning of a word among provided in the *sense inventory* was used in the context. The sense inventory may be obtained by a WSI system or provided as external information.

Many natural language processing (NLP) applications benefit from ability to deal with word ambiguity (Navigli & Crisafulli, 2010; Vickrey et al., 2005). Since word representations have been used as word features in dependency parsing (Chen & Manning, 2014), named-entity recognition (Turian et al., 2010) and sentiment analysis (Maas et al., 2011) among many other tasks, employing multi-prototype representations could increase the performance of such representation-based approaches.

In this paper we develop natural extension of the Skip-gram

Appearing in Proceedings of the 19^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2016, Cadiz, Spain. JMLR: W&CP volume 51.

model which we call Adaptive Skip-gram (AdaGram). It retains all noticeable properties of SG such as fast online learning and high quality of representations while allowing to automatically learn the necessary number of prototypes per word at desired *semantic resolution*.

The rest of the paper is organized as follows: we start with reviewing original Skip-gram model (section 2), then we describe our extension called Adaptive Skip-gram (section 3). Then, we compare our model to existing approaches in section 4. In section 5 we evaluate our model qualitatively by considering neighborhoods of selected words in the learned latent space and by quantitative comparison against concurrent approaches. Finally, we conclude in section 6.

2 Skip-gram model

The original Skip-gram model (Mikolov et al., 2013a) is formulated as a set of grouped word prediction tasks. Each task consists of prediction of a word v given a word w using correspondingly their *output* and *input* representations

$$p(v|w,\theta) = \frac{\exp(in_w^\intercal out_v)}{\sum_{v'=1}^{V}\exp(in_w^\intercal out_{v'})}, \qquad (1)$$

where global parameter $\theta = \{in_v, out_v\}_{v=1}^{V}$ stands for both input and output representations for all words of the dictionary indexed with $1,\dots,V$. Both input and output representations are real vectors of the dimensionality D.

These individual predictions are grouped in a way to simultaneously predict *context words* $\mathbf{y}$ of some input word x:

$$p(\mathbf{y}|x,\theta) = \textstyle\prod_j p(y_j|x,\theta).$$

Input text $\mathbf{o}$ consisting of N words $o_1, o_2, \dots, o_N$ is then interpreted as a sequence of input words $X = \{x_i\}_{i=1}^{N}$ and their contexts $Y = \{\mathbf{y}_i\}_{i=1}^{N}$. Here i-th training object (x_i, $\mathbf{y}_i$) consists of word $x_i = o_i$ and its context $\mathbf{y}_i = \{o_t\}_{t\in c(i)}$ where $c(i)$ is a set of indices such that $|t-i| \le C/2$ and $t \neq i$ for all $t \in c(i)$[1]!

Finally, Skip-gram objective function is the likelihood of contexts given the corresponding input words:

$$p(Y|X,\theta) = \prod_{i=1}^{N} p(\mathbf{y}_i|x_i,\theta) = \prod_{i=1}^{N}\prod_{j=1}^{C} p(y_{ij}|x_i,\theta). \qquad (2)$$

Note that although contexts of adjacent words intersect, the model assumes the corresponding prediction problems independent.

For training the Skip-gram model it is common to ignore sentence and document boundaries and to interpret the input data as a stream of words. The objective (2) is then optimized in a stochastic fashion by sampling i-th word and its context, estimating gradients and updating parameters θ. After the model is trained, Mikolov et al. (2013a) treated the input representations of the trained model as word features and showed that they captured semantic similarity between concepts represented by the words. Further we refer to the input representations as *prototypes* following (Reisinger & Mooney, 2010b).

Both evaluation and differentiation of (1) has linear complexity (in the dictionary size V) which is too expensive for practical applications. Because of that the soft-max prediction model (1) is substituted by the hierarchical soft-max (Mnih & Hinton, 2008):

$$p(v|w,\theta) = \textstyle\prod_{n\in path(v)} \sigma(ch(n) in_w^\intercal out_n). \qquad (3)$$

Here output representations are no longer associated with words, but rather with nodes in a binary tree where leaves are all possible words in the dictionary with unique paths from root to corresponding leaf. $ch(n)$ assigns either 1 or -1 to each node in the $path(v)$ depending on whether n is a left or right child of previous node in the path. Equation (3) is guaranteed to sum to 1 i.e. be a distribution w.r.t. v as $\sigma(x) = 1/(1+\exp(-x)) = 1-\sigma(-x)$. For computational efficiency Skip-gram uses Huffman tree to construct hierarchical soft-max.

3 Adaptive Skip-gram

The original Skip-gram model maintains only one prototype per word. It would be unrealistic to assume that single representation may capture the semantics of all possible word meanings. At the same time it is non-trivial to specify exactly the right number of prototypes required for handling meanings of a particular word. Hence, an adaptive approach for allocation of additional prototypes for ambiguous words is required. Further we describe our Adaptive Skip-gram (AdaGram) model which extends the original Skip-gram and may automatically learn the required number of prototypes for each word using Bayesian nonparametric approach.

First, assume that each word has K meanings each associated with its own prototype. That means that we have to modify (3) to account for particular choice of the meaning. For this reason we introduce latent variable z that encodes the index of active meaning and extend (3) to $p(v|z=k,w,\theta) = \prod_{n\in path(v)} \sigma(ch(n) in_{wk}^\intercal out_n)$. Note that we bring even more asymmetry between input and output representations compared to (3) since now only prototypes depend on the particular word meaning. While it is possible to make context words be also meaning-aware this would make the training process much more complicated. Our experiments show that this word prediction model is enough to capture word ambiguity. This could be viewed as prediction of context *words* using *meanings* of the input words.

[1] For notational simplicity we will further assume that size of the context is always equal to C which is true for all non-boundary words.

However, setting the number of prototypes for all words equal is not a very realistic assumption. Moreover, it is desirable that the number of prototypes for a particular word would be determined by the training text corpus. We approach this problem by employing Bayesian nonparametrics into Skip-gram model, i.e. we use the constructive definition of Dirichlet process (Ferguson, 1973) for automatic determination of the required number of prototypes. Dirichlet process (DP) has been successfully used for infinite mixture modeling and other problems where the number of structure components (e.g. clusters, latent factors, etc.) is not known a priori which is exactly our case.

We use the constructive definition of DP via the stick-breaking representation (Sethuraman, 1994) to define a prior over meanings of a word. The meaning probabilities are computed by dividing total probability mass into infinite number of diminishing pieces summing to 1. So the prior probability of k-th meaning of the word w is

$$p(z = k|w, \boldsymbol{\beta}) = \beta_{wk} \prod_{r=1}^{k-1} (1 - \beta_{wr}),$$
$$p(\beta_{wk}|\alpha) = \mathrm{Beta}(\beta_{wk}|1, \alpha), \quad k = 1, \dots$$

This assumes that infinite number of prototypes for each word may exist. However, as long as we consider finite amount of text data, the number of prototypes (those with non-zero prior probabilities) for word w will not exceed the number of occurrences of w in the text which we denote as n_w. The hyperparameter α controls the number of prototypes for a word allocated a priori. Asymptotically, the expected number of prototypes of word w is proportional to $\alpha \log(n_w)$. Thus, larger values of α produce more prototypes which lead to more granular and specific meanings captured by learned representations and the number of prototypes scales logarithmically with number of occurrences.

Another attractive property of DPs is their ability to increase the complexity of latent variables' space with more data arriving. In our model this will result to more distinctive meanings of words discovered on larger text corpus.

Combining all parts together we may write the AdaGram model as follows:

$$p(Y, Z, \boldsymbol{\beta}|X, \alpha, \theta) = \prod_{w=1}^{V} \prod_{k=1}^{\infty} p(\beta_{wk}|\alpha)$$
$$\prod_{i=1}^{N} \Big[p(z_i|x_i, \boldsymbol{\beta}) \prod_{j=1}^{C} p(y_{ij}|z_i, x_i, \theta) \Big],$$

where $Z = \{z_i\}_{i=1}^N$ is a set of senses for all the words. Similarly to Mikolov et al. (2013a) we do not consider any regularization (and so the informative prior) for representations and seek for point estimate of θ.

3.1 Learning representations

One way to train the AdaGram is to maximize the marginal likelihood of the model

$$\log p(Y|X, \theta, \alpha) = \log \int \sum_{Z} p(Y, Z, \boldsymbol{\beta}|X, \alpha, \theta) d\beta \quad (4)$$

with respect to representations θ. One may see that the marginal likelihood is intractable because of the latent variables Z and $\boldsymbol{\beta}$. Moreover, $\boldsymbol{\beta}$ and θ are infinite-dimensional parameters. Thus unlike the original Skip-gram and other methods for learning multiple word representations, our model could not be straightforwardly trained by stochastic gradient ascent w.r.t. θ.

To make this tractable we consider the variational lower bound on the marginal likelihood (4)

$$\mathcal{L} = \mathbb{E}_q \left[\log p(Y, Z, \boldsymbol{\beta}|X, \alpha, \theta) - \log q(Z, \boldsymbol{\beta}) \right]$$

where $q(Z, \boldsymbol{\beta}) = \prod_{i=1}^{N} q(z_i) \prod_{w=1}^{V} \prod_{k=1}^{T} q(\beta_{wk})$ is the fully factorized variational approximation to the posterior $p(Z, \boldsymbol{\beta}|X, Y, \alpha, \theta)$ with possible number of representations for each word truncated to T (Blei & Jordan, 2005). It may be shown that the maximization of the variational lower bound with respect to $q(Z, \boldsymbol{\beta})$ is equivalent to the minimization of Kullback-Leibler divergence between $q(Z, \boldsymbol{\beta})$ and the true posterior (Jordan et al., 1999).

Within this approximation the variational lower bound $\mathcal{L}(q(Z), q(\boldsymbol{\beta}), \theta)$ takes the following form:

$$\mathcal{L}(q(Z), q(\boldsymbol{\beta}), \theta) = \mathbb{E}_q \Bigg[\sum_{w=1}^{V} \sum_{k=1}^{T} \log p(\beta_{wk}|\alpha) - \log q(\beta_{wk}) +$$
$$\sum_{i=1}^{N} \big(\log p(z_i|x_i, \boldsymbol{\beta}) - \log q(z_i) + \sum_{j=1}^{C} \log p(y_{ij}|z_i, x_i, \theta) \big) \Bigg].$$

Setting derivatives of $\mathcal{L}(q(Z), q(\boldsymbol{\beta}), \theta)$ with respect to $q(Z)$ and $q(\boldsymbol{\beta})$ to zero yields standard update equations

$$\log q(z_i = k) = \mathbb{E}_{q(\boldsymbol{\beta})} \Big[\log \beta_{x_i,k} + \sum_{r=1}^{k-1} \log(1 - \beta_{x_i,r}) \Big]$$
$$+ \sum_{j=1}^{C} \log p(y_{ij}|k, x_i, \theta) + \text{const}, \quad (5)$$

$$\log q(\boldsymbol{\beta}) = \sum_{w=1}^{V} \sum_{k=1}^{T} \log \mathrm{Beta}(\beta_{wk}|a_{wk}, b_{wk}), \quad (6)$$

where (natural) parameters a_{wk} and b_{wk} deterministically depend on the expected number of assignments to particular sense $n_{wk} = \sum_{i:x_i=w} q(z_i = k)$ (Blei & Jordan, 2005): $a_{wk} = 1 + n_{wk}, b_{wk} = \alpha + \sum_{r=k+1}^{T} n_{wr}$.

Stochastic variational inference. Although variational updates given by (5) and (6) are tractable, they require the full pass over training data. In order to keep the efficiency of Skip-gram training procedure, we employ stochastic variational inference approach (Hoffman et al., 2013) and derive online optimization algorithm for the maximization of $\mathcal{L}$. There are two groups of parameters in our objective: $\{q(\beta_{vk})\}$ and θ are global because they affect all the objects; $\{q(z_i)\}$ are local, i.e. affect only the corresponding object x_i. After updating the local parameters according to (5) with the global parameters fixed and defining the

obtained distribution as $q^*(Z)$ we have a function of the global parameters

$$\mathcal{L}^*(q(\boldsymbol{\beta}), \theta) = \mathcal{L}(q^*(Z), q(\boldsymbol{\beta}), \theta) \geq \mathcal{L}(q(Z), q(\boldsymbol{\beta}), \theta).$$

The new lower bound $\mathcal{L}^*$ is no longer a function of local parameters which are always kept updated to their optimal values. Following Hoffman et al. (2013) we iteratively optimize $\mathcal{L}^*$ with respect to the global parameters using stochastic gradient estimated at a single object. Stochastic gradient w.r.t θ computed on the i-th object is computed as follows:

$$\widehat{\nabla}_\theta \mathcal{L}^* = N \sum_{j=1}^{C} \sum_{k=1}^{T} q^*(z_i = k) \nabla_\theta \log p(y_{ij} | k, x_i, \theta).$$

Now we describe how to optimize $\mathcal{L}^*$ w.r.t global posterior approximation $q(\boldsymbol{\beta}) = \prod_{w=1}^{D} \prod_{k=1}^{T} q(\beta_{wk})$. The stochastic gradient with respect to natural parameters a_{wk} and b_{wk} according to (Hoffman et al., 2013) can be estimated by computing intermediate values of natural parameters $(\hat{a}_{wk}, \hat{b}_{wk})$ on the i-th data point as if we estimated $q(z)$ for all occurrences of $x_i = w$ equal to $q(z_i)$:

$$\hat{a}_{wk} = 1 + n_w q(z_i = k), \hat{b}_{wk} = \alpha + \sum_{r=k+1}^{T} n_w q(z_i = r),$$

where n_w is the total number of occurrences of word w. The stochastic gradient estimate then can be expressed in the following simple form:

$$\widehat{\nabla}_{a_{wk}} \mathcal{L}^* = \widehat{a}_{wk} - a_{wk}, \widehat{\nabla}_{b_{wk}} \mathcal{L}^* = \widehat{b}_{wk} - b_{wk}.$$

One may see that making such gradient update is equivalent to updating counts n_{wk} since they are sufficient statistics of $q(\beta_{wk})$.

We use conservative initialization strategy for $q(\boldsymbol{\beta})$ starting with only one allocated meaning for each word, i.e. $n_{w1} = n_w$ and $n_{wk} = 0, k > 1$. Representations are initialized with random values drawn from $\mathrm{Uniform}(-0.5/D, 0.5/D)$. In our experiments we updated both learning rates ρ and λ using the same linear schedule from 0.025 to 0.

The resulting learning algorithm 1 may be also interpreted as an instance of stochastic variational EM algorithm. It has linear computational complexity in the length of text $\mathbf{o}$ similarly to Skip-gram learning procedure. The overhead of maintaining variational distributions is negligible comparing to dealing with representations and thus training of AdaGram is T times slower than Skip-gram.

3.2 Disambiguation and prediction

After model is trained on data $\mathcal{D} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$, it can be used to infer the meanings of an input word x given its context $\mathbf{y}$. The predictive probability of a meaning can be computed as

$$p(z = k | x, \mathcal{D}, \theta, \alpha) \propto \int p(z = k | \boldsymbol{\beta}, x) q(\boldsymbol{\beta}) d\boldsymbol{\beta}, \quad (7)$$

Algorithm 1 Training AdaGram model

Input: training data $\{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$, hyperparameter α
Output: parameters θ, distributions $q(\boldsymbol{\beta})$, $q(\boldsymbol{z})$

```
Initialize parameters θ, distributions q(β), q(z)
for i = 1 to N do
    Select word w = x_i and its context y_i
    LOCAL STEP:
    for k = 1 to T do
        γ_ik = E_{q(β_w)}[log p(z_i = k | β, x_i)]
        for j = 1 to C do
            γ_ik ← γ_ik + log p(y_ij | x_i, k, θ)
        end
    end
    γ_ik ← exp(γ_ik) / Σ_ℓ exp(γ_iℓ)
    GLOBAL STEP:
    ρ_t ← 0.025(1 − i/N), λ_t ← 0.025(1 − i/N)
    for k = 1 to T do
        Update n_wk ← (1 − λ_t) n_wk + λ_t n_w γ_ik
    end
    Update θ ← θ + ρ_t ∇_θ Σ_k Σ_j γ_ik log p(y_ij | x_i, k, θ)
end
```

where $q(\boldsymbol{\beta})$ can serve as an approximation of $p(\boldsymbol{\beta} | \mathcal{D}, \theta, \alpha)$. Since $q(\boldsymbol{\beta})$ has the form of independent Beta distributions whose parameters are given in sec. 3.1 the integral can be taken analytically. The number of learned prototypes for a word w may be computed as $\sum_{k=1}^{T} \mathbb{1}[p(z = k | w, \mathcal{D}, \theta, \alpha) > \epsilon]$ where ϵ is a threshold e.g. 10^{-3}.

The probability of each meaning of x given context $\mathbf{y}$ is thus given by

$$p(z = k | x, \mathbf{y}, \theta) \propto p(\mathbf{y} | x, k, \theta) \int p(k | \boldsymbol{\beta}, x) q(\boldsymbol{\beta}) d\boldsymbol{\beta} \quad (8)$$

Now the posterior predictive over context words $\mathbf{y}$ given input word x may be expressed as

$$p(\mathbf{y} | x, \mathcal{D}, \theta, \alpha) = \int \sum_{z=1}^{T} p(\mathbf{y} | x, z, \theta) p(z | \boldsymbol{\beta}, x) q(\boldsymbol{\beta}) d\boldsymbol{\beta}. \quad (9)$$

4 Related work

Literature on learning continuous-space representations of embeddings for words is vast, therefore we concentrate on works that are most relevant to our approach.

In the works (Huang et al., 2012; Reisinger & Mooney, 2010a,b) various neural network-based methods for learning multi-prototype representations are proposed. These methods include clustering contexts for all words as prepossessing or intermediate step. While this allows to learn multiple prototypes per word, clustering large number of contexts brings serious computational overhead and limit these approaches to offline setting.

Recently various modifications of Skip-gram were proposed to learn multi-prototype representations. Proximity-

Ambiguity Sensitive Skip-gram (Qiu et al., 2014) maintains individual representations for different parts of speech (POS) of the same word. While this may handle word ambiguity to some extent, clearly there could be many meanings even for the same part of speech of some word remaining not discovered by this approach.

Work of Tian et al. (2014) can be considered as a parametric form of our model with number of meanings for each word fixed. Their model also provides improvement over original Skip-gram, but it is not clear how to set the number of prototypes. Our approach not only allows to efficiently learn required number of prototypes for ambiguous words, but is able also to gradually increase the number of meanings when more data becomes available thus distinguishing between shades of same meaning.

It is also possible to incorporate external knowledge about word meanings into Skip-gram in the form of sense inventory (Chen et al., 2014). First, single-prototype representations are pre-trained with original Skip-gram. Afterwards, meanings provided by WordNet lexical database are used learn multi-prototype representations for ambiguous words. The dependency on the external high-quality linguistic resources such as WordNet makes this approach inapplicable to languages lacking such databases. In contrast, our model does not consider any form of supervision and learns the sense inventory automatically from the raw text.

Recent work of Neelakantan et al. (2014) proposing Multi-sense Skip-gram (MSSG) and its nonparameteric (not in the sense of Bayesian nonparametrics) version (NP MSSG) is the closest to AdaGram prior art. While MSSG defines the number of prototypes a priori similarly to (Tian et al., 2014), NP MSSG features automatic discovery of multiple meanings for each word. In contrast to our approach, learning for NP MSSG is defined rather as ad-hoc greedy procedure that allocates new representation for a word if existing ones explain its context below some threshold. AdaGram instead follows more principled nonparametric Bayesian approach.

5 Experiments

In this section we empirically evaluate our model in a number of different tests. First, we demonstrate learned multi-prototype representations on several example words. We investigate how different values of α affect the number of learned prototypes what we call a semantic resolution of a model. Then we evaluate our approach on the word sense induction task (WSI). We also provide more experiments in the supplementary material.

In order to evaluate our method we trained several models with different values of α on April 2010 snapshot of English Wikipedia (Shaoul & Westbury, 2010). It contains nearly 2 million articles and 990 million tokens. We did not consider words which have less than 20 occurrences. The context width was set to $C = 10$ and the truncation level of Stick-breaking approximation (the maximum number of meanings) to $T = 30$. The dimensionality D of representations learned by our model was set to 300 to match the dimensionality of the models we compare with.

Table 1: Nearest neighbors of meaning prototypes learned by the AdaGram model with $\alpha = 0.1$. In the second column we provide the predictive probability of each meaning.

WORD	$p(z)$	NEAREST NEIGHBOURS
python	0.33	monty, spamalot, cantsin
	0.42	perl, php, java, c++
	0.25	molurus, pythons
apple	0.34	almond, cherry, plum
	0.66	macintosh, iifx, iigs
date	0.10	unknown, birth, birthdate
	0.28	dating, dates, dated
	0.31	to-date, stateside
	0.31	deadline, expiry, dates
bow	0.46	stern, amidships, bowsprit
	0.38	spear, bows, wow, sword
	0.16	teign, coxs, evenlode
mass	0.22	vespers, masses, liturgy
	0.42	energy, density, particle
	0.36	wholesale, widespread
run	0.02	earned, saves, era
	0.35	managed, serviced
	0.26	2-run, ninth-inning
	0.37	drive, go, running, walk
net	0.34	pre-tax, pretax, billion
	0.28	negligible, total, gain
	0.16	fox, est/edt, sports
	0.23	puck, ball, lobbed
fox	0.38	cbs, abc, nbc, espn
	0.14	raccoon, wolf, deer, foxes
	0.33	abc, tv, wonderfalls
	0.14	gardner, wright, taylor
rock	0.23	band, post-hardcore
	0.10	little, big, arkansas
	0.29	pop, funk, r&b, metal, jazz
	0.14	limestone, bedrock
	0.23	'n', roll, 'n', 'n

5.1 Nearest neighbours of learned prototypes

In Table 1 we present the meanings which were discovered by our model with parameter $\alpha = 0.1$ for words used in (Neelakantan et al., 2014) and for a few other sample words. To distinguish the meanings we obtain their nearest neighbors by computing the cosine similarity between each meaning prototype and the prototypes of meanings of all other words. One may see that AdaGram model learns a reasonable number of prototypes which are meaningful and interpretable. The predictive probability of each meaning reflects how frequently it was used in the training corpus.

For most of the words $\alpha = 0.1$ results in most interpretable model. It seems that for values less than 0.1 for most words only one prototype is learned and for values greater than 0.1 the model becomes less interpretable as learned meanings are too specific sometimes duplicating.

Table 2: Nearest neighbours of different prototypes of words "light" and "core" learned by AdaGram under different values of α and corresponding predictive probabilities.

ALPHA	$p(z)$	nearest neighbours
		"light"
Skip-Gram	1.00	far-red, emitting
0.075	0.28	armoured, amx-13, kilcrease
	0.72	bright, sunlight, luminous
0.1	0.09	tvärbanan, hudson-bergen
	0.17	dark, bright, green
	0.09	4th, dragoons, 2nd
	0.26	radiation, ultraviolet
	0.28	darkness, shining, shadows
	0.11	self-propelled, armored
		"core"
Skip-Gram	1.00	cores, components, i7
0.075	0.3	competencies, curriculum
	0.34	cpu, cores, i7, powerxcell
	0.36	nucleus backbone
0.1	0.21	reactor, hydrogen-rich
	0.13	intel, processors
	0.27	curricular, competencies
	0.15	downtown, cores, center
	0.24	nucleus, rag-tag, roster

5.2 Semantic resolution

As mentioned in section 3 hyperparameter α of the AdaGram model indirectly controls the number of induced word meanings. Figure 1, Left shows the distribution of number of induced word meanings under different values of α. One may see that while for most words relatively small number of meanings is learned, larger values of α lead to more meanings in general. This effect may be explained by the property of Dirichlet process to allocate number of prototypes that logarithmically depends on number of word occurrences. Since word occurrences are known to be distributed by Zipf's law, the majority of words is rather infrequent and thus our model discovers few meanings for them. Figure 1, Right quantitatively demonstrates this phenomenon.

In the Table 2 we demonstrate how larger values of α lead to more meanings on the example of the word "light". The original Skip-gram discovered only the meaning related to a physical phenomenon, AdaGram with $\alpha = 0.075$ found the second, military meaning, with further increase of α value those meanings start splitting to submeanings, e.g. light tanks and light troops. Similar results are provided for the word "core".

5.3 Word prediction

Since both Skip-gram and AdaGram are defined as models for predicting context of a word, it is essential to evaluate how well they explain test data by predictive likelihood. We use last 200 megabytes of December 2014 snapshot of English Wikipedia as test data for this experiment.

Table 3: Test log-likelihood under different α on sample from Wikipedia (see sec. 5.3) and Adjusted Rand Index (ARI) on the training part of WWSI dataset (see sec. 5.4).

MODEL	LOG-LIKELIHOOD	ARI
Skip-Gram.300D	-7.403	-
Skip-Gram.600D	-7.387	-
AdaGram.300D $\alpha = 0.05$	-7.399	0.007
AdaGram.300D $\alpha = 0.1$	-7.385	0.226
AdaGram.300D $\alpha = 0.15$	-7.382	**0.268**
AdaGram.300D $\alpha = 0.2$	-7.378	0.254
AdaGram.300D $\alpha = 0.25$	**-7.375**	0.250
AdaGram.300D $\alpha = 0.5$	-7.387	0.230

Similarly to the train procedure we consider this text as pairs of input words and contexts of size $C = 10$, that is, $\mathcal{D}_{test} = \{(x_i, \mathbf{y}_i)\}_{i=1}^{N}$ and compare AdaGram with original Skip-gram by average log-likelihood (see sec. 3.2). We were unable to include MSSG and NP-MSSG into the comparison as these models do not estimate conditional word likelihood. The results are given in Table 3. Clearly, AdaGram models text data better than Skip-gram under wide range of values of α.

Since AdaGram has more parameters than Skip-Gram with the same dimensionality of representations, it is natural to compare its efficiency with Skip-Gram that has the comparable number of parameters. The model with $\alpha = 0.15$ which we study extensively further has approximately 2 learned prototypes per word in average, so we doubled the dimensionality of Skip-Gram and included it into comparison as well[2]. One may see that AdaGram with α equal to 0.15 outperforms 600-dimensional Skip-Gram and so does the model with $\alpha = 0.1$.

5.4 Word-sense induction

The nonparametric learning of a multi-prototype representation model is closely related to the word-sense induction (WSI) task which aims at automatic discovery of different meanings for the words. Indeed, learned prototypes identify different word meanings and it is natural to assess how well they are aligned with human judgements.

We compare our AdaGram model with Nonparametric Multi-sense Skip-gram (NP-MSSG) proposed by Neelakantan et al. (2014) which is currently the only existing approach to learning multi-prototype word representations with Skip-gram. We also include in comparison the parametric form of NP-MSSG which has the number of meanings fixed to 3 for all words during the training. All models were trained on the same dataset which is the Wikipedia snapshot by Shaoul & Westbury (2010). For the comparison with MSSG and NP-MSSG we used source code and

[2]Also note that 600-dimensional Skip-Gram has twice more parameters in hierarchical softmax than 300-dimensional AdaGram

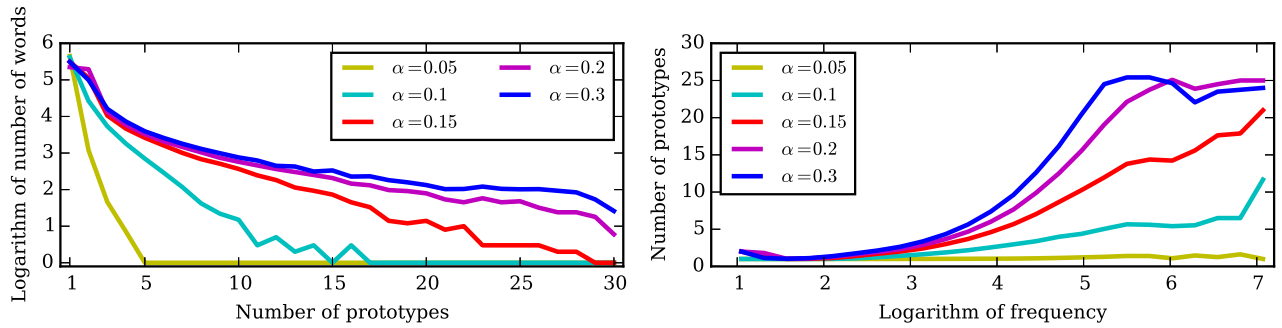

Figure 1: Left: Distribution of number of word meanings learned by AdaGram model for different values of parameter α. For the number of meanings k we plot the $\log_{10}(n_k + 1)$, where n_k is the number of words with k meanings. Right: All words in dictionary were divided into 30 bins according to the logarithm of their frequency. Here we plot the number of learned prototypes averaged over each such bin.

Table 4: Adjusted rand index (ARI) for word sense induction task for different datasets. Here we use the test subset of WWSI dataset. See sec. 5.4 for details.

MODEL	SE-2007	SE-2010	SE-2013	WWSI
MSSG.300D.30K	0.048	0.085	0.033	0.194
NP-MSSG.50D.30K	0.031	0.058	0.023	0.163
NP-MSSG.300D.6K	0.033	0.044	0.033	0.110
MPSG.300D	0.044	0.077	0.014	0.160
AdaGram.300D.$\alpha = 0.15$	**0.069**	**0.097**	**0.061**	**0.286**

models released by the authors. Neelakantan et al. (2014) limited the number of words for which multi-prototype representations were learned (30000 and 6000 most frequent words) for these models. We use the following notation: 300D or 50D is the dimensionality of word representations, 6K or 30K is the number of multi-prototype words (6000 and 30000 respectively) in case of MSSG and NP-MSSG models. Another baseline is Multi-prototype Skip-Gram (MPSG) proposed by Tian et al. (2014) which in contrast to AdaGram has uniform distribution over fixed number of senses. We have trained this model similarly to (Tian et al., 2014) setting number of senses for each word equal to 3.

The evaluation is performed as follows. Dataset consisting of *target word* and *context* pairs is supplied to a model which uses the context to disambiguate target word into a meaning from its learned sense inventory. Then for each target word the model's labeling of contexts and ground truth one are compared as two different clusterings of the same set using appropriate metrics. The results are then averaged over all target words.

Data We consider several WSI datasets in our experiments. The SemEval-2007 dataset was introduced for SemEval-2007 Task 2 competition, it contains 27232 contexts collected from Wall Street Journal (WSJ) corpus. The SemEval-2010 was similarly collected for the SemEval-2010 Task 14 competition and contains 8915 contexts in total, part obtained from web pages returned by a search engine and the other part from news articles. We also consider SemEval-2013 Task 13 dataset consisting from 4664 contexts (we considered only single-term words in this dataset).

In order to make the evaluation more comprehensive, we introduce the new Wikipedia Word-sense Induction (WWSI) dataset consisting of 188 target words and 36354 contexts. For the best of our knowledge it is currently the largest WSI dataset available. While SemEval datasets are prepared with hand effort of experts which mapped contexts into gold standard sense inventory, we collected WWSI using fully automatic approach from December 2014 snapshot of Wikipedia. The dataset is splitted evenly into train and test parts. More details on the dataset construction procedure are provided in the supplementary material, sec. 3.

For all SemEval datasets we merged together train and test contexts and used them for model comparison. Each model was supplied with contexts of the size that maximizes its ARI performance.

Metrics Authors of SemEval dataset Manandhar et al. (2010) suggested two metrics for model comparison: V-Measure (VM) and F-Score (FS). They pointed to the weakness of both VM and FS. VM favours large number of clusters and attains large values on unreasonable clusterings which assign each instance to its own cluster while FS is biased towards clusterings consisting of small number of clusters e.g. assigning each instance to the same single cluster. Thus we consider another metric - adjusted Rand index (ARI) (Hubert & Arabie, 1985) which does not suffer from such drawbacks. Both examples of undesirable clus-

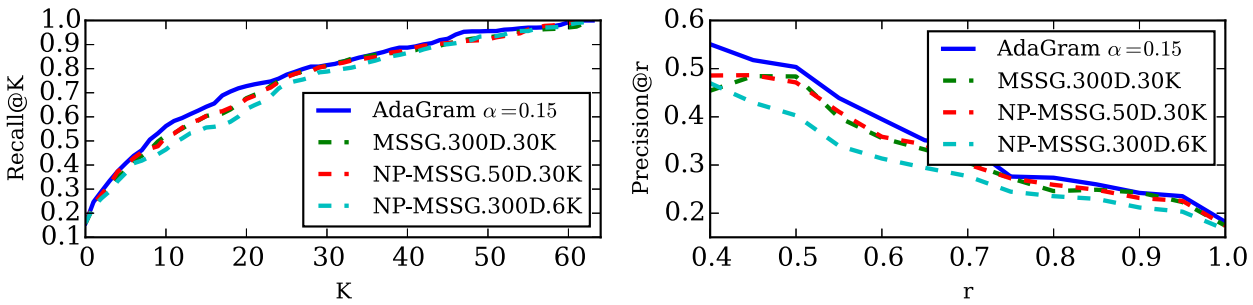

Figure 2: Recall (left) and precision (right) for task-11 of Semeval-2013 competition. K is the position of the snippet in the search result page, r is the recall value, see text for details.

terings described above will get ARI of nearly zero which corresponds to human intuition. Thus we consider ARI as more reliable metric for WSI evaluation. We still report VM and FS values in the suppl. material (sec. 4) in order to make our results comparable to others obtained on the datasets.

Evaluation Since AdaGram is essentially influenced by hyperparameter α, we first investigate how different choices of α affect WSI performance on the train part of our WWSI dataset in terms of ARI, see table 3. The model with $\alpha = 0.15$ attains maximum ARI and thus we use this model for all further experiments.

We compare AdaGram against MSSG and NP-MSSG on all datasets described above, see table 4 for results. AdaGram consistently outperforms the concurrent approaches on all datasets and achieves significant improvement on the test part of WWSI dataset. One may see that nonparametric version of MSSG delivers consistently worse performance than MSSG with number of prototypes fixed to 3. This suggests that the ability to discover different word meanings is rather limited for NP-MSSG. The fact that AdaGram substantially outperforms NP-MSSG indicates that more principled Bayesian nonparametric approach is more suitable for the task of word-sense induction.

One may note that results on SemEval datasets are smaller than results on WWSI dataset consistently for all models. We explain this by the difference between the train corpus and test data used for preparing SemEval datasets such as news articles as well as by the difference between sense inventories, i.e. SemEval data uses WordNet and OntoNotes as sources of word meanings.

5.5 Web search results diversification

In this experiment we follow the methodology of Semeval-2013 Task 11 competition. Systems are given an ambiguous query and web search result snippets which have to be clustered. The main goal of this task is to measure the ability of systems to diversify web search results. The authors of this task (Di Marco & Navigli, 2013) proposed to use two following metrics for evalutation. Subtopic Recall@K measures how many different word meanings (from the gold standard sense inventory) are covered by top K diversified search results. Subtopic Precision@r determines the ratio of different meanings provided in the first K_r results where K_r is minumum number of top K results achieving recall r. We consider only single-token words in this comparison. The results are shown in the Figure 2. One may see that curves of AdaGram are monotonically higher than curves of all concurrent models suggesting that AdaGram is more suitable on such real-world application.

6 Conclusion

In the paper we proposed AdaGram which is the Bayesian nonparametric extension of the well-known Skip-gram model. AdaGram uses different prototypes to represent a word depending on the context and thus may handle various forms of word ambiguity. Our experiments suggest that representations learned by our model correspond to different word meanings. Using resolution parameter α we may control how many prototypes are extracted from the same text corpus. Too large values of α lead to different prototypes that correspond to the same meaning which decreases model performance. The values $\alpha = 0.1 - 0.2$ are generally good for practical purposes. For those values the truncation level $T = 30$ is enough and does not affect the number of discovered prototypes. AdaGram also features online variational learning algorithm which is very scalable and makes it possible to train our model just several times slower than extremely efficient Skip-gram model. Since the problem of learning multi-prototype word representation is closely related to word-sense induction, we evaluated AdaGram on several WSI datasets and contributed a new large one obtained automatically from Wikipedia disambiguation pages. The source code of our implementation, WWSI dataset and all trained models are available at `https://github.com/sbos/AdaGram.jl`.

Acknowledgements

The work was supported by RBFR grants 15-31-20596, 14-01-31361 and by Microsoft Azure for Research Award. Anton Osokin was supported by the MSR-INRIA Joint Center.

References

Blei, D. M. and Jordan, M. I. Variational inference for Dirichlet process mixtures. *Bayesian Analysis*, 1:121–144, 2005.

Chen, D. and Manning, C. A fast and accurate dependency parser using neural networks. In *EMNLP*, pp. 740–750, 2014.

Chen, X., Liu, Z., and Sun, M. A unified model for word sense representation and disambiguation. In *EMNLP*, pp. 1025–1035, 2014.

Di Marco, Antonio and Navigli, Roberto. Clustering and diversifying web search results with graph-based word sense induction. *Computational Linguistics*, 39(3):709–754, 2013.

Ferguson, T. S. A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *JMLR*, 14(1):1303–1347, 2013.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. Improving word representations via global context and multiple word prototypes. In *ACL*, 2012.

Hubert, L. and Arabie, P. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. Learning word vectors for sentiment analysis. In *NAACL HLT*, pp. 142–150, 2011.

Manandhar, S., Klapaftis, I. P., Dligach, D., and Pradhan, S. S. SemEval-2010 task 14: Word sense induction & disambiguation. In *SemEval*, pp. 63–68, 2010.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NIPS*, pp. 3111–3119, 2013a.

Mikolov, T., Yih, W., and Zweig, G. Linguistic regularities in continuous space word representations. In *NAACL HLT*, pp. 746–751, 2013b.

Mnih, A. and Hinton, G. E. A scalable hierarchical distributed language model. In *NIPS*, pp. 1081–1088, 2008.

Navigli, R. and Crisafulli, G. Inducing word senses to improve web search result clustering. In *EMNLP*, pp. 116–126, 2010.

Neelakantan, A., Shankar, J., Passos, A., and McCallum, A. Efficient non-parametric estimation of multiple embeddings per word in vector space. In *EMNLP*, 2014.

Qiu, L., Cao, Y., Nie, Z., Yu, Y., and Rui, Y. Learning word representation considering proximity and ambiguity. In *AAAI*, pp. 1572–1578, 2014.

Reisinger, J. and Mooney, R. A mixture model with sharing for lexical semantics. In *EMNLP*, pp. 1173–1182, 2010a.

Reisinger, J. and Mooney, R. J. Multi-prototype vector-space models of word meaning. In *NAACL HLT*, pp. 109–117, 2010b.

Sethuraman, J. A constructive definition of Dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

Shaoul, C. and Westbury, C. The Westbury lab Wikipedia corpus. 2010.

Tian, F., Dai, H., Bian, J., Gao, B., Zhang, R., Chen, E., and Liu, T.-Y. A probabilistic model for learning multi-prototype word embeddings. In *COLING*, pp. 151–160, 2014.

Turian, J., Ratinov, L., and Bengio, Y. Word representations: A simple and general method for semi-supervised learning. In *ACL*, pp. 384–394, 2010.

Vickrey, D., Biewald, L., Teyssier, M., and Koller, D. Word-sense disambiguation for machine translation. In *EMNLP*, 2005.