Low-Rank Regularization for Sparse Conjunctive Feature Spaces: An Application to Named Entity Classification

Audi Primadhanty
Universitat Politècnica de Catalunya
primadhanty@cs.upc.edu

Xavier Carreras **Ariadna Quattoni**
Xerox Research Centre Europe
xavier.carreras@xrce.xerox.com
ariadna.quattoni@xrce.xerox.com

Abstract

Entity classification, like many other important problems in NLP, involves learning classifiers over sparse high-dimensional feature spaces that result from the conjunction of elementary features of the entity mention and its context. In this paper we develop a low-rank regularization framework for training max-entropy models in such sparse conjunctive feature spaces. Our approach handles conjunctive feature spaces using matrices and induces an implicit low-dimensional representation via low-rank constraints. We show that when learning entity classifiers under minimal supervision, using a seed set, our approach is more effective in controlling model capacity than standard techniques for linear classifiers.

1 Introduction

Many important problems in NLP involve learning classifiers over sparse high-dimensional feature spaces that result from the conjunction of elementary features. For example, to classify an entity in a document, it is standard to exploit features of the left and right context in which the entity occurs as well as spelling features of the entity mention itself. These sets of features can be grouped into vectors which we call elementary feature vectors. In our example, there will be one elementary feature vector for the left context, one for the right context and one for the features of the mention. Observe that, when the elementary vectors consist of binary indicator features, the outer product of any pair of vectors represents all conjunctions of the corresponding elementary features.

Ideally, we would like to train a classifier that can leverage all conjunctions of elementary features, since among them there might be some that are discriminative for the classification task at hand. However, allowing for such expressive high dimensional feature space comes at a cost: data sparsity becomes a key challenge and controlling the capacity of the model is crucial to avoid overfitting the training data.

The problem of data sparsity is even more severe when the goal is to train classifiers with minimal supervision, i.e. small training sets. For example, in the entity classification setting we might be interested in training a classifier using only a small set of examples of each entity class. This is a typical scenario in an industrial setting, where developers are interested in classifying entities according to their own classification schema and can only provide a handful of examples of each class.

A standard approach to control the capacity of a linear classifier is to use ℓ_1 or ℓ_2 regularization on the parameter vector. However, this type of regularization does not seem to be effective when dealing with sparse conjunctive feature spaces. The main limitation is that ℓ_1 and ℓ_2 regularization can not let the model give weight to conjunctions that have not been observed at training. Without such ability it is unlikely that the model will generalize to novel examples, where most of the conjunctions will be unseen in the training set.

Of course, one could impose a strong prior on the weight vector so that it assigns weight to unseen conjunctions, but how can we build such a prior? What kind of reasonable constraints can we put on unseen conjunctions?

Another common approach to handle high dimensional conjunctive feature spaces is to manually design the feature function so that it includes only a subset of "relevant" conjunctions. But designing such a feature function can be time consuming and one might need to design a new feature function for each classification task. Ideally, we would have a learning algorithm that does not require such feature engineering and that it can automatically leverage rich conjunctive feature spaces.

In this paper we present a solution to this problem by developing a regularization framework specifically designed for sparse conjunctive feature spaces. Our approach results in a more effective way of controlling model capacity and it does not require feature engineering.

Our strategy is based on:

- Employing tensors to define the scoring function of a max-entropy model as a multilinear form that computes weighted inner products between elementary vectors.
- Forcing the model to induce low-dimensional embeddings of elementary vectors via low-rank regularization on the tensor parameters.

The proposed regularization framework is based on a simple conceptual trick. The standard approach to handle conjunctive feature spaces in NLP is to regard the parameters of the linear model as long vectors computing an inner product with a high dimensional feature representation that lists explicitly all possible conjunctions. Instead, the parameters of our the model will be tensors and the compatibility score between an input pattern and a class will be defined as the sum of multilinear functions over elementary vectors.

We then show that the rank[1] of the tensor has a very natural interpretation. It can be seen as the intrinsic dimensionality of a latent embedding of the elementary feature vectors. Thus by imposing a low-rank penalty on the tensor parameters we are encouraging the model to induce a low-dimensional projection of the elementary feature vectors . Using the rank itself as a regularization constraint in the learning algorithm would result in a non-convex optimization. Instead, we follow a standard approach which is to use the nuclear norm as a convex relaxation of the rank.

In summary the main contributions of this paper are:

- We develop a new regularization framework for training max-entropy models in high-dimensional sparse conjunctive feature spaces. Since the proposed regularization implicitly induces a low dimensional embedding of feature vectors, our algorithm can also be seen as a way of implicitly learning a latent variable model.
- We present a simple convex learning algorithm for training the parameters of the model.
- We conduct experiments on learning entity classifiers with minimal supervision. Our results show that the proposed regularization framework is better for sparse conjunctive feature spaces than standard ℓ_2 and ℓ_1 regularization. These results make us conclude that encouraging the max-entropy model to operate on a low-dimensional space is an effective way of controlling the capacity of the model an ensure good generalization.

2 Entity Classification with Log-linear Models

The formulation we develop in this paper applies to any prediction task whose inputs are some form of tuple. We focus on classification of entity mentions, or entities in the context of a sentence. Formally, our input objects are tuples $x = \langle l, e, r \rangle$ consisting of an entity e, a left context l and a right context r. The goal is to classify x into one entity class in the set $\mathcal{Y}$.

We will use log-linear models of the form:

$$\Pr(y \mid x; \theta) = \frac{\exp\{s_\theta(x, y)\}}{\sum_{y'} \exp\{s_\theta(x, y')\}} \qquad (1)$$

where $s_\theta : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is a scoring function of entity tuples with a candidate class, and θ are the parameters of this function, to be specified below.

In the literature it is common to employ a feature-based linear model. That is, one defines a feature function $\phi : \mathcal{X} \to \{0, 1\}^n$ that represents entity tuples in an n-dimensional binary feature space[2], and the model has a weight vector for each class, $\theta = \{\mathbf{w}_y\}_{y \in \mathcal{Y}}$. Then $s_\theta(x, y) = \phi(x) \cdot \mathbf{w}_y$.

[1]There are many ways of defining the rank of a tensor. In this paper we *matricize* tensors into matrices and use the rank of the resulting matrix. Matricization is also referred to as unfolding.

[2]In general, all models in this paper accept real-valued feature functions. But we focus on binary *indicator* features because in practice these are the standard type of features in NLP classifiers, and the ones we use here. In fact, in this paper we develop feature spaces based on *products* of elementary feature functions, in which case the resulting representations correspond to conjunctions of the elementary features.

3 Low-rank Entity Classification Models

In this section we propose a specific family of models for classifying entity tuples.

3.1 A Low-rank Model of Left-Right Contexts

We start from the observation that when representing tuple objects such as $x = \langle l, e, r \rangle$ with features, we often depart from a feature representation of each element of the tuple. Hence, let ϕ_{l} and ϕ_{r} be two feature functions representing left and right contexts, with binary dimensions d_1 and d_2 respectively. For now, we will define a model that ignores the entity mention e and makes predictions using context features. It is natural to define conjunctions of left and right features. Hence, in its most general form, one can define a matrix $\mathbf{W}_y \in \mathbb{R}^{d_1 \times d_2}$ for each class, such that $\theta = \{\mathbf{W}_y\}_{y \in \mathcal{Y}}$ and the score is:

$$s_\theta(\langle l, e, r \rangle, y) = \phi_{\mathrm{l}}(l)^\top \mathbf{W}_y \phi_{\mathrm{r}}(r) \quad . \tag{2}$$

Note that this corresponds to a feature-based linear model operating in the product space of ϕ_{l} and ϕ_{r}, that is, the score has one term for each pair of features: $\sum_{i,j} \phi_{\mathrm{l}}(l)[i]\ \phi_{\mathrm{r}}(r)[j]\ \mathbf{W}_y[i,j]$. Note also that it is trivial to include elementary features of ϕ_{l} and ϕ_{r}, in addition to conjunctions, by having a constant dimension in each of the two representations set to 1.

In all, the model in Eq. (2) is very expressive, with the caveat that it can easily overfit the data, specially when we work only with a handful of labeled examples. The standard way to control the capacity of a linear model is via ℓ_1 or ℓ_2 regularization.

Regarding our parameters as matrices allows us to control the capacity of the model via regularizers that favor parameter matrices with low rank. To see the effect of these regularizers, consider that $\mathbf{W}_y$ has rank k, and let $\mathbf{W}_y = \mathbf{U}_y \mathbf{\Sigma}_y \mathbf{V}_y^\top$ be the singular value decomposition, where $\mathbf{U}_y \in \mathbb{R}^{d_1 \times k}$ and $\mathbf{V}_y \in \mathbb{R}^{d_2 \times k}$ are orthonormal projections and $\mathbf{\Sigma}_y \in \mathbb{R}^{k \times k}$ is a diagonal matrix of singular values. We can rewrite the score function as

$$s_\theta(\langle l, e, r \rangle, y) = (\phi_{\mathrm{l}}(l)^\top \mathbf{U}_y)\ \mathbf{\Sigma}_y\ (\mathbf{V}_y^\top \phi_{\mathrm{r}}(r)) \quad . \tag{3}$$

In words, the rank k is the *intrinsic dimensionality* of the inner product behind the score function. A low-rank regularizer will favor parameter matrices that have low intrinsic dimensionality. Below we describe a convex optimization for low-rank models using *nuclear norm* regularization.

3.2 Adding Entity Features

The model above classifies entities based only on the context. Here we propose an extension to make use of features of the entity. Let $\mathcal{T}$ be a set of possible entity feature tags, i.e. tags that describe an entity, such as IsCapitalized, ContainsDigits, SingleToken, ... Let ϕ_{e} be a feature function representing entities. For this case, to simplify our expression, we will use a set notation and denote by $\phi_{\mathrm{e}}(e) \subseteq \mathcal{T}$ the set of feature tags that describe e. Our model will be defined with one parameter matrix per feature tag and class label, i.e. $\theta = \{\mathbf{W}_{t,y}\}_{t \in \mathcal{T}, y \in \mathcal{Y}}$. The model form is:

$$s_\theta(\langle l, e, r \rangle, y) = \sum_{t \in \phi_{\mathrm{e}}(e)} \phi_{\mathrm{l}}(l)^\top \mathbf{W}_{t,y} \quad .\phi_{\mathrm{r}}(r). \tag{4}$$

3.3 Learning with Low-rank Constraints

In this section we describe a convex procedure to learn models of the above form that have low rank. We will define an objective that combines a loss and a regularization term.

Our first observation is that our parameters are a tensor with up to four axes, namely left and right context representations, entity features, and entity classes. While a matrix has a clear definition of rank, it is not the case for general tensors, and there exist various definitions in the literature. The technique that we use is based on *matricization* of the tensor, that is, turning the tensor into a matrix that has the same parameters as the tensor but organized in two axes. This is done by partitioning the tensor axes into two sets, one for matrix rows and another for columns. Once the tensor has been turned into a matrix, we can use the standard definition of matrix rank. A main advantage of this approach is that we can make use of standard routines like singular value decomposition (SVD) to decompose the matricized tensor. This is the main reason behind our choice.

In general, different ways of partitioning the tensor axes will lead to different notions of intrinsic dimensions. In our case we choose the left context axes as the row dimension, and the rest of axes as the column dimension.[3] In this section, we will

[3]In preliminary experiments we tried variations, such as having right prefixes in the columns, and left prefixes, entity tags and classes in the rows. We only observer minor, non-significant variations in the results.

denote as $\mathbf{W}$ the matricized version of the parameters θ of our models.

The second observation is that minimizing the rank of a matrix is a non-convex problem. We make use of a convex relaxation based on the nuclear norm (Srebro and Shraibman, 2005). The *nuclear norm*[4] of a matrix $\mathbf{W}$, denoted $\|\mathbf{W}\|_\star$, is the sum of its singular values: $\|\mathbf{W}\|_\star = \sum_i \mathbf{\Sigma}_{i,i}$ where $\mathbf{W} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ is the singular value decomposition of $\mathbf{W}$. This norm has been used in several applications in machine learning as a convex surrogate for imposing low rank, e.g. (Srebro et al., 2004).

Thus, the nuclear norm is used as a regularizer. With this, we define our objective as follows:

$$\underset{\mathbf{W}}{\operatorname{argmin}} \, L(\mathbf{W}) + \tau R(\mathbf{W}) \quad , \tag{5}$$

where $L(\mathbf{W})$ is a convex loss function, $R(\mathbf{W})$ is a regularizer, and τ is a constant that trades off error and capacity. In experiments we will compare nuclear norm regularization with ℓ_1 and ℓ_2 regularizers. In all cases we use the negative log-likelihood as loss function, denoting the training data as $\mathcal{D}$:

$$L(\mathbf{W}) = \sum_{(\langle l,e,r\rangle,y)\in\mathcal{D}} -\log \Pr(y \mid \langle l,e,r\rangle; \mathbf{W}) \quad . \tag{6}$$

To solve the objective in Eq. (5) we use a simple optimization scheme known as *forward-backward splitting (FOBOS)* (Duchi and Singer, 2009). In a series of iterations, this algorithm performs a gradient update followed by a proximal projection of the parameters. Such projection depends on the regularizer used: for ℓ_1 it thresholds the parameters; for ℓ_2 it scales them; and for nuclear-norm regularization it thresholds the singular values. This means that, for nuclear norm regularization, each iteration requires to decompose $\mathbf{W}$ using SVD. See (Madhyastha et al., 2014) for details about this optimization for a related application.

4 Related Work

The main aspect of our approach is the use of a spectral penalty (i.e., the rank) to control the capacity of multilinear functions parameterized by matrices or tensors. Quattoni et al. (2014) used nuclear-norm regularization to learn latent-variable max-margin sequence taggers. Madhyastha et al. (2014) defined bilexical distributions parameterized by matrices which result lexical embeddings tailored for a particular linguistic relation. Like in our case, the low-dimensional latent projections in these papers are learned implicitly by imposing low-rank constraints on the predictions of the model.

Lei et al. (2014) also use low-rank tensor learning in the context of dependency parsing, where like in our case dependencies are represented by conjunctive feature spaces. While the motivation is similar, their technical solution is different. We use the technique of matricization of a tensor combined with a nuclear-norm relaxation to obtain a convex learning procedure. In their case they explicitly look for a low-dimensional factorization of the tensor using a greedy alternating optimization.

Also recently, Yao et al. (2013) have framed entity classification as a low-rank matrix completion problem. The idea is based on the fact that if two entities (in rows) have similar descriptions (in columns) they should have similar classes. The low-rank structure of the matrix defines intrinsic representations of entities and feature descriptions. The same idea was applied to relation extraction (Riedel et al., 2013), using a matrix of entity pairs times descriptions that corresponds to a matricization of an entity-entity-description tensor. Very recently Singh et al. (2015) explored alternative ways of applying low-rank constraints to tensor-based relation extraction.

Another aspect of this paper is training entity classification models using minimal supervision, which has been addressed by multiple works in the literature. A classical successful approach for this problem is to use co-training (Blum and Mitchell, 1998): learn two classifiers that use different views of the data by using each other's predictions. In the same line, Collins and Singer (1999) trained entity classifiers by bootstraping from an initial set of seeds, using a boosting version of co-training. Seed sets have also been exploited by graphical model approaches. Haghighi and Klein (2006) define a graphical model that is soft-constrained such that the prediction for an unlabeled example agrees with the labels of seeds that are distributionally similar. Li et al. (2010) present a Bayesian approach to expand an initial seed set, with the goal of creating a gazetteer.

Another approach to entity recognition that, like in our case, learns projections of contextual features is the method by Ando and Zhang (2005).

[4]Also known as the trace norm.

Class		Nb Mentions			
	10-30 Seed	**10-30**	**40-120**	**640-1920**	**All**
PER	clinton, dole, arafat, yeltsin, wasim akram, lebed, dutroux, waqar younis, mushtaq ahmed, croft	334	747	3,133	6,516
LOC	u.s., england, germany, britain, australia, france, spain, pakistan, italy, china	1,384	2,885	5,812	6,159
ORG	reuters, u.n., oakland, puk, osce, cincinnati, eu, nato, ajax, honda	295	699	3,435	5,271
MISC	russian, german, british, french, dutch, english, israeli, european, iraqi, australian	611	1326	3,085	3,205
O	year, percent, thursday, government, police, results, tuesday, soccer, president, monday, friday, people, minister, sunday, division, week, time, state, market, years, officials, group, company, saturday, match, at, world, home, august, standings	5,326	11,595	31,071	36,673

Table 1: For each entity class, the seed of entities for the **10-30** set, together with the number of mentions in the training data that involve entities in the seed for various sizes of the seeds.

They define a set of auxiliary tasks, which can be supervised using unlabeled data, and find a projection of the data that works well as input representation for the auxiliary tasks. This representation is then used for the target task.

More recently Neelakantan and Collins (2014) presented another approach to gazetteer expansion using an initial seed. A novel aspect is the use of Canonical Correlation Analysis (CCA) to compute embeddings of entity contexts, that are used by the named entity classifier. Like in our case, their method learns a compressed representation of contexts that helps prediction.

5 Experiments

In this section we evaluate our regularization framework for training models in high-dimensional sparse conjunctive feature spaces. We run experiments on learning entity classifiers with minimal supervision. We focus on classification of unseen entities to highlight the ability of the regularizer to generalize over conjunctions that are not observed at training. We simulate minimal supervision using the CoNLL-2003 Shared Task data (Tjong Kim Sang and De Meulder, 2003), and compare the performance to ℓ_1 and ℓ_2 regularizers.

5.1 Minimal Supervision Task

We use a minimal supervision setting where we provide the algorithm a seed of entities for each class, that is, a list of entities that is representative for that class. The assumption is that any mention of an entity in the seed is a positive example for the corresponding class. Given unlabeled data and a seed of entities for each class, the goal is to learn a model that correctly classifies mentions of entities that are not in the seed. In addition to standard entity classes, we also consider a special non-entity class, which is part of the classification but is excluded from evaluation.

Note that named entity classification for *unseen* entities is a challenging problem. Even in the standard fully-supervised scenario, when we measure the performance of state-of-the-art methods on unseen entities, the F1 values are in the range of 60%. This represents a significant drop with respect to the standard metrics for named entity recognition, which consider all entity mentions of the test set irrespective of whether they appear in the training data or not, and where F1 values at 90% levels are obtained (e.g. (Ratinov and Roth, 2009)). This suggests that part of the success of state-of-the-art models is in storing known entities together with their type (in the form of gazetteers or directly in lexicalized parameters of the model).

5.2 Setting

We use the CoNLL-2003 English data, which is annotated with four types: person (PER), location (LOC), organization (ORG), and miscellaneous (MISC). In addition, the data is tagged with parts-of-speech (PoS), and we compute word clusters running the Brown clustering algorithm (Brown et al., 1992) on the words in the training set.

We consider annotated entity phrases as candidate entities, and all single nouns that are not part of an entity as candidate non-entities (O). Both candidate entities and non-entities will be referred to as candidates in the remaining of this section. We lowercase all candidates and remove the am-

Features	Window	Bag-of-words		N-grams	
		Lexical	Cluster	Lexical	Cluster
Elementary features of left and right contexts	1	13.63	**14.59**	13.63	**14.59**
	2	**15.49**	13.86	13.08	**13.54**
	3	12.18	**14.45**	12.14	**13.28**
Only full conjunctions of left and right contexts	1	12.90	**13.75**	12.90	**13.75**
	2	8.59	**8.85**	12.31	**12.43**
	3	8.57	**10.59**	10.15	**10.49**
Elementary features and all conjunctions of left and right contexts	1	15.30	**16.98**	15.30	**16.98**
	2	**13.26**	12.89	14.28	**15.33**
	3	**11.87**	11.54	**13.94**	**13.15**

Table 2: Average-F1 of classification of unseen entity candidates on development data, using the **10-30** training seed and ℓ_2 regularization, for different conjunctive spaces (elementary only, full conjunctions, all). **Bag-of-words** elementary features contain all clusters/PoS in separate windows to the left and to the right of the candidate. **N-grams** elementary features contain all n-grams of clusters/PoS in separate left and right windows (e.g. for size 3 it includes unigrams, bigrams and trigrams on each side).

biguous ones (i.e., those with more than one label in different mentions).[5]

To simulate a minimal supervision, we create supervision seeds by picking the n most frequent training candidates for entity types, and the m most frequent candidate non-entities. We create seeds of various sizes n-m, namely **10-30**, **40-120**, **640-1920**, as well as **all** of the candidates. For each seed, the training set consists of all training mentions that involve entities in the seed. Table 1 shows the smaller seed, as well as the number of mentions for each seed size.

For evaluation we use the development and test sections of the data, but we remove the instances of candidates in the training data (i.e., that are in the **all** seed). We do not remove instances that are ambiguous in the tests. [6] As evaluation metric we use the average F1 score computed over all entity types, excluding the non-entity type.

[5]In the CoNLL-2003 English training set, only 235 candidates are ambiguous out of 13,441 candidates, i.e. less than 2%. This suggests that in this data the difficulty behind the task is in recognizing and classifying unseen entities, and not in disambiguating known entities in a certain context.

[6]After removing the ambiguous candidates from the training data, and removing candidates seen in the training from the development and test sets, this is the number of mentions (and number of unique candidates in parenthesis) in the data used in our experiments:

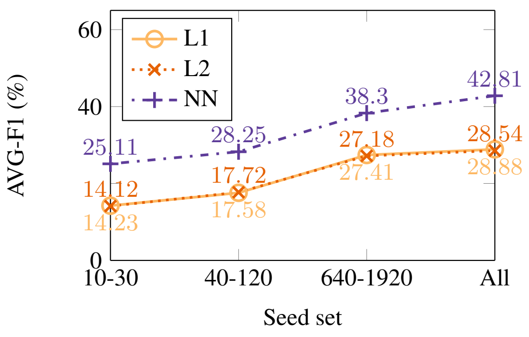
	training	dev.	test
PER	6,516 (3,489)	1,040 (762)	1,342 (925)
LOC	6,159 (987)	176 (128)	246 (160)
ORG	5,271 (2,149)	400 (273)	638 (358)
MISC	3,205 (760)	177 (142)	213 (152)
O	36,673 (5,821)	951 (671)	995 (675)

5.3 Context Representations

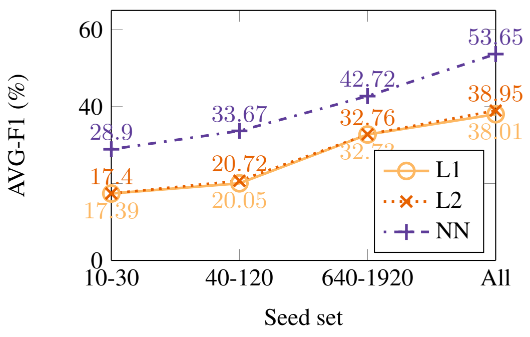
We refer to context as the sequence of tokens before (left context) and after (right context) a candidate mention in a sentence. Different classifiers can be built using different representations of the contexts. For example we can change the window size of the context sequence (i.e., for a window size of 1 we only use the last token before the mention and the first token after the mention). We can treat the left and right contexts independently of each other, we can treat them as a unique combination, or we can use both. We can also choose to use the word form of a token, its PoS tag, a word cluster, or a combination of these.

Table 2 compares different context representations and their performance in classifying unseen candidates using maximum-entropy classifiers trained with Mallet (McCallum, 2002) with ℓ_2 regularization, using the **10-30** seed. We use the lexical representation (the word itself) and a word cluster representation of the context tokens and use a window size of one to three. We use two types of features: bag-of-words features (1-grams of tokens in the specified window) and n-gram features (with n smaller or equal to the window size). The performance of using word clusters is comparable, and sometimes better, to using lexical representations. Moreover, using a longer window, in this case, does not necessarily result in better performance. [7] In the rest of the experiments

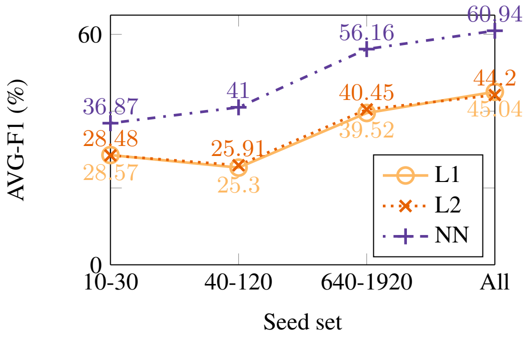
[7]Our learner and feature configuration, using ℓ_2 regularization, obtains state-of-the-art results on the standard evalu-



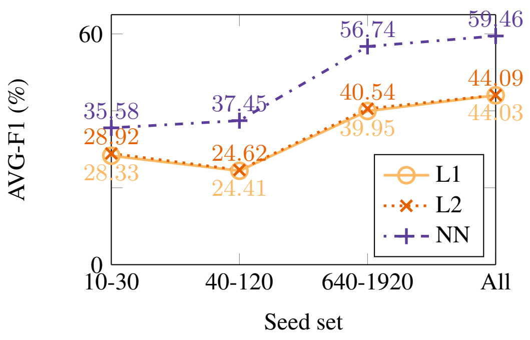
(a) Only full conjunctions of left-right contexts (cluster), window size = 1



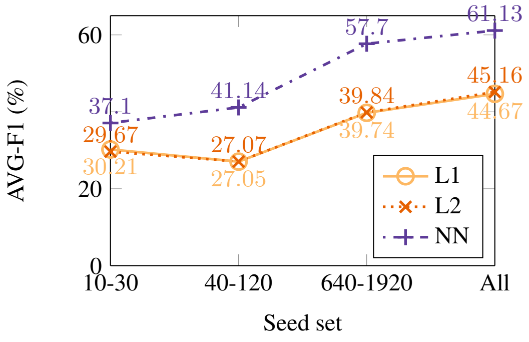
(b) Only full conjunctions of entity tags and left-right contexts (cluster), window size = 1



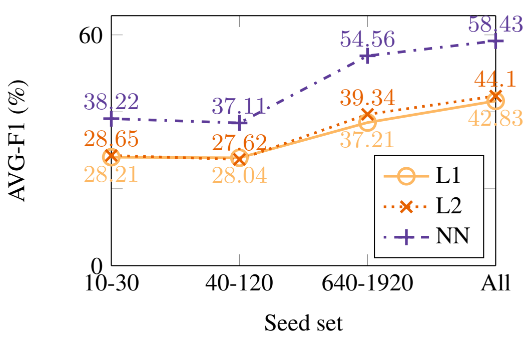
(c) Elementary features and all conjunctions of entity tags and left-right contexts (cluster), window size = 1



(d) Elementary features and all conjunctions of entity tags and left-right contexts (cluster), window size = 2



(e) Elementary features and all conjunctions of entity tags and left-right contexts (cluster & PoS), window size = 1



(f) Elementary features and all conjunctions of entity tags and left-right contexts (cluster & PoS), window size = 2

Figure 1: Average F1 of classification of unseen entity candidates on development data, with respect to the size of the seed. NN refers to models with nuclear norm regularization, L1 and L2 refer to ℓ_1 and ℓ_2 regularization. Each plot corresponds to a different conjunctive feature space with respect to window size (1 or 2), context representation (cluster with/out PoS), using entity features or not, and combining or not full conjunctions with lower-order conjunctions and elementary features.

- **cap=1**, **cap=0**: whether the first letter of the entity candidate is uppercase, or not
- **all-low=1**, **all-low=0**: whether all letters of the candidate are lowercase letters, or not
- **all-cap1=1**, **all-cap1=0**: whether all letters of the candidate are uppercase letters, or not
- **all-cap2=1**, **all-cap2=0**: whether all letters of the candidate are uppercase letters and periods, or not
- **num-tokens=1**, **num-tokens=2**, **num-tok>2**: whether the candidate consists of one token, two or more
- **dummy**: a tag that holds for any entity candidate, used to capture context features alone

Table 3: The 12 entity tags used to represent entity candidates. The tags **all-cap1** and **all-cap2** are from (Neelakantan and Collins, 2014).

		PER			LOC			ORG			MISC			AVG F1
		PREC	REC	F1	PREC	REC	F1	PREC	REC	F1	PREC	REC	F1	
10-30	ℓ_1	65.69	65.40	65.55	**15.38**	**23.58**	**18.62**	59.33	19.44	**29.28**	23.36	30.05	26.28	34.93
	ℓ_1	65.54	64.80	65.17	15.12	23.17	18.30	**60.82**	18.50	28.37	23.30	30.52	26.42	34.56
	NN	**72.41**	**74.52**	**73.45**	14.89	21.55	17.61	49.09	**21.16**	17.61	**31.40**	**38.03**	**34.40**	**38.76**
40-120	ℓ_1	72.16	44.07	54.72	13.38	40.24	20.08	48.89	31.19	38.09	22.03	35.68	27.24	35.03
	ℓ_2	71.75	44.89	55.23	**13.61**	**41.87**	**20.54**	**49.39**	31.50	38.47	21.64	30.99	25.48	34.93
	NN	**75.16**	**61.33**	**67.54**	13.08	20.73	16.04	49.03	**35.74**	**41.34**	**29.97**	**47.42**	**36.73**	**40.41**
640-1920	ℓ_1	79.52	62.27	69.85	23.59	**44.31**	30.79	55.78	47.65	51.39	19.81	30.05	23.88	43.98
	ℓ_2	78.62	65.55	71.49	26.55	43.50	32.97	**60.19**	49.06	**54.06**	21.73	31.92	25.86	46.10
	NN	**80.73**	**80.55**	**80.64**	**51.91**	**44.31**	**47.81**	53.82	**54.08**	53.95	**29.14**	**51.17**	**37.14**	**54.88**
All	ℓ_1	75.58	72.48	74.00	32.84	36.18	34.43	57.28	46.24	51.17	27.93	29.11	28.51	47.03
	ℓ_2	**76.59**	70.77	73.57	34.21	**36.99**	35.55	57.79	**50.00**	53.61	28.93	32.86	30.77	48.37
	NN	73.83	**90.84**	**81.46**	**64.96**	36.18	**46.48**	**72.11**	44.98	**55.41**	**37.20**	**43.66**	**40.17**	**55.88**

Table 4: Results on the test for models trained with different sizes of the seed, using the parameters and features that obtain the best evaluation results the development set. NN refers to nuclear norm regularization, L1 and L2 refer to ℓ_1 and ℓ_2 regularization. Only test entities unseen at training are considered. Avg. F1 is over PER, LOC, ORG and MISC, excluding O.

we will use the elementary features that are more predictive and compact: clusters and PoS tags in windows of size at most 2.

5.4 Comparing Regularizers

We compare the performance of models trained using the nuclear norm regularizer with models trained using ℓ_1 and ℓ_2 regularizers. To train each model, we validate the regularization parameter and the number of iterations on development data, trying a wide range of values. The best performing configuration is then used for the comparison.

Figure 1 shows results on the development set for different feature sets. We started representing context using cluster labels, as it is the most compact representation obtaining good results in preliminary experiments. We tried several conjunctions: a conjunction of the left and right context, as well as conjunctions of left and right contexts and features of the candidate entity. We also tried all different conjunction combinations of the contexts and the candidate entity features, as well as adding PoS tags to represent contexts. To represent an entity candidate we use standard traits of the spelling of the mention, such as capitalization, the existence of symbols, as well as the number of tokens in the candidate. See Table 3 for the definition of the features describing entity candidates.

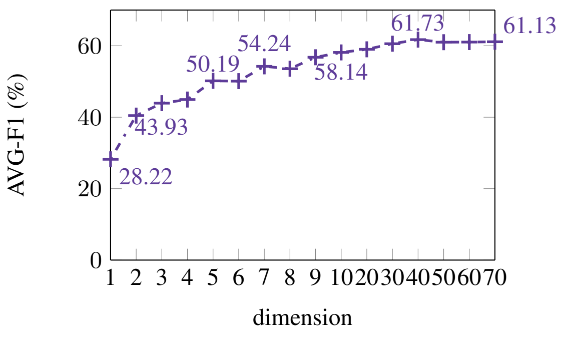
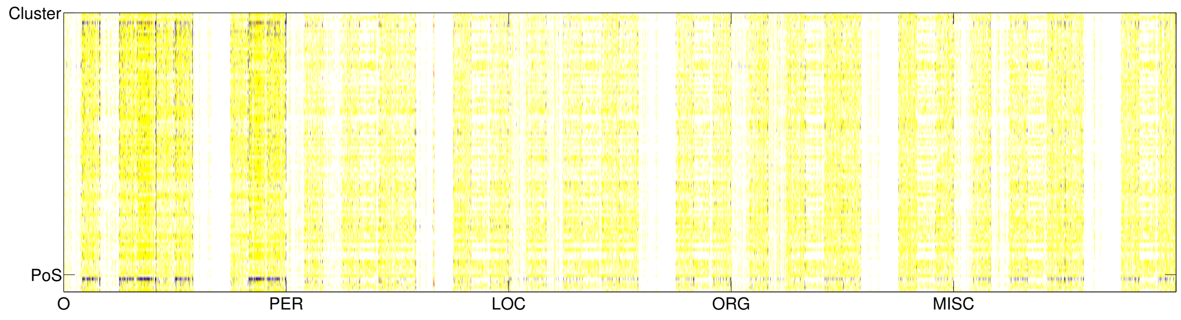


Figure 2: Avg. F1 on development for increasing dimensions, using the low-rank model in Figure 1e trained with **all** seeds.

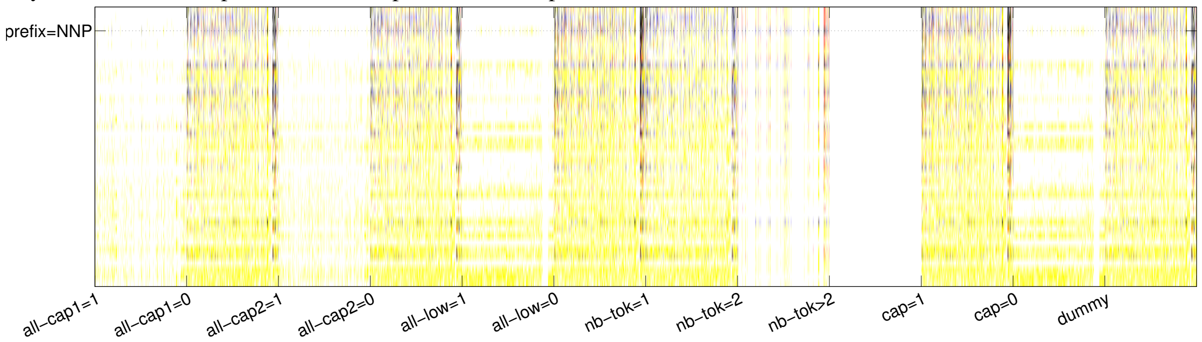
We observe that for most conjunction settings our regularizer performs better than the ℓ_1 and ℓ_2 regularizers. Using the best model from each regularizer, we evaluated on the test set. Table 4 shows the test results. For all seed sets, the nuclear norm regularizer obtains the best average F1 performance. This shows that encouraging the max-entropy model to operate on a low-dimensional space is effective. Moreover, Figure 2 shows model performance as a function of the number of dimensions of the intrinsic projection. The model obtains a good performance even if only a few intrinsic dimensions are used.

Figure 3 shows the parameter matrix of the low-

ation. Using our richest feature set, the model obtains 76.76 of accuracy in the development, for the task of classifing entities with correct boundaries. If we add features capturing the full entity and its tokens, then the accuracy is 87.63, which is similar to state-of-the-art performance (the best results in literature typically exploit additional gazetteers). Since our evaluation focuses on unknown entities, our features do not include information about the word tokens of entites.



(a) Full parameter matrix of the low-rank model. The ticks in x-axis indicate the space for different entity types, while the ticks in y-axis indicate the space for different prefix context representations.



(b) The subblock for PER entity type and PoS representation of the prefixes. The ticks in x-axis indicate the space of the entity features used, while the tick in y-axis indicates an example of a frequently observed prefix for this entity type.

Figure 3: Parameter matrix of the low-rank model in Figure 1f trained with the **10-30** seed, with respect to observations of the associated features in training and development. Non-white conjunctions correspond to non-zero weights: black is for conjunctions seen in both the training and development sets; blue is for those seen in training but not in the development; red indicates that the conjunctions were observed only in the development; yellow is for those not observed in training nor development.

rank model in Figure 1f trained with the **10-30** seed, with respect to observed features in training and development data. Many of the conjunctions of the development set were never observed in the training set. Our regularizer framework is able to propagate weights from the conjunctive features seen in training to unseen conjunctive features that are close to each other in the projected space (these are the yellow and red cells in the matrix). In contrast, ℓ_1 and ℓ_2 regularization techniques can not put weight on unseen conjunctions.

6 Conclusion

We have developed a low-rank regularization framework for training max-entropy models in sparse conjunctive feature spaces. Our formulation is based on using tensors to parameterize classifiers. We control the capacity of the model using the nuclear-norm of a matricization of the tensor. Overall, our formulation results in a convex procedure for training model parameters.

We have experimented with these techniques in the context of learning entity classifiers. Compared to ℓ_1 and ℓ_2 penalties, the low-rank model obtains better performance, without the need to manually specify feature conjunctions. In our analysis, we have illustrated how the low-rank approach can assign non-zero weights to conjunctions that were unobserved at training, but are similar to observed conjunctions with respect to the low-dimensional projection of their elements.

We have used matricization of a tensor to define its rank, using a fixed transformation of the tensor into a matrix. Future work should explore how to combine efficiently different transformations.

Acknowledgements

We thank Gabriele Musillo and the anonymous reviewers for their helpful comments and suggestions. This work has been partially funded by the Spanish Government through the SKATER project (TIN2012-38584-C06-01) and an FPI predoctoral grant for Audi Primadhanty.

References

Rie Kubota Ando and Tong Zhang. 2005. A framework for learning predictive structures from multiple tasks and unlabeled data. *J. Mach. Learn. Res.*, 6:1817–1853, December.

Avrim Blum and Tom Mitchell. 1998. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92–100, New York, NY, USA. ACM.

Peter F. Brown, Peter V. deSouza, Robert L. Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479.

Michael Collins and Yoram Singer. 1999. Unsupervised models for named entity classification. In *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*.

John Duchi and Yoram Singer. 2009. Efficient online and batch learning using forward backward splitting. *Journal of Machine Learning Research*, 10:2899–2934.

Aria Haghighi and Dan Klein. 2006. Prototype-driven learning for sequence models. In *Proceedings of the Main Conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 320–327, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, Maryland, June. Association for Computational Linguistics.

Xiao-Li Li, Lei Zhang, Bing Liu, and See-Kiong Ng. 2010. Distributional similarity vs. pu learning for entity set expansion. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 359–364, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pranava Swaroop Madhyastha, Xavier Carreras, and Ariadna Quattoni. 2014. Learning Task-specific Bilexical Embeddings. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 161–171, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Andrew K. McCallum. 2002. Mallet: A machine learning for language toolkit.

Arvind Neelakantan and Michael Collins. 2014. Learning dictionaries for named entity recognition using minimal supervision. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 452–461, Gothenburg, Sweden, April. Association for Computational Linguistics.

Ariadna Quattoni, Borja Balle, Xavier Carreras, and Amir Globerson. 2014. Spectral regularization for max-margin sequence tagging. In Tony Jebara and Eric P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1710–1718. JMLR Workshop and Conference Proceedings.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June. Association for Computational Linguistics.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 74–84, Atlanta, Georgia, June. Association for Computational Linguistics.

Sameer Singh, Tim Rocktäschel, and Sebastian Riedel. 2015. Towards Combined Matrix and Tensor Factorization for Universal Schema Relation Extraction. In *NAACL Workshop on Vector Space Modeling for NLP (VSM)*.

Nathan Srebro and Adi Shraibman. 2005. Rank, trace-norm and max-norm. In *Learning Theory*, pages 545–560. Springer Berlin Heidelberg.

Nathan Srebro, Jason Rennie, and Tommi S Jaakkola. 2004. Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

Limin Yao, Sebastian Riedel, and Andrew McCallum. 2013. Universal schema for entity type prediction. In *Proceedings of the 2013 Workshop on Automated Knowledge Base Construction*, AKBC '13, pages 79–84, New York, NY, USA. ACM.