Comparing Convolutional Neural Networks to Traditional Models for Slot Filling

Heike Adel and **Benjamin Roth** and **Hinrich Schütze**
Center for Information and Language Processing (CIS)
LMU Munich
Oettingenstr. 67, 80538 Munich, Germany
heike@cis.lmu.de

Abstract

We address relation classification in the context of slot filling, the task of finding and evaluating fillers like "Steve Jobs" for the slot X in "X founded Apple". We propose a convolutional neural network which splits the input sentence into three parts according to the relation arguments and compare it to state-of-the-art and traditional approaches of relation classification. Finally, we combine different methods and show that the combination is better than individual approaches. We also analyze the effect of genre differences on performance.

1 Introduction

Structured knowledge about the world is useful for many natural language processing (NLP) tasks, such as disambiguation, question answering or semantic search. However, the extraction of structured information from natural language text is challenging because one relation can be expressed in many different ways. The TAC Slot Filling (SF) Shared Task defines slot filling as extracting fillers for a set of predefined relations ("slots") from a large corpus of text data. Exemplary relations are the city of birth of a person or the employees or founders of a company. Participants are provided with an evaluation corpus and a query file consisting of pairs of entities and slots. For each entity slot pair (e.g. "Apple" and "founded_by"), the systems have to return the second argument ("filler") of the relation (e.g. "Steve Jobs") as well as a supporting sentence from the evaluation corpus. The key challenge in slot filling is relation classification: given a sentence s of the evaluation corpus containing the name of a queried entity (e.g., "Apple") and a filler candidate (e.g., "Steve Jobs"), we need to decide whether s expresses the relation ("founded_by", in this case). We will refer to the mentions of the two arguments of the relation as *name* and *filler*. Performance on relation classification is crucial for slot filling since its effectiveness directly depends on it.

In this paper, we investigate three complementary approaches to relation classification.

The first approach is pattern matching, a leading approach in the TAC evaluations. Fillers are validated based on patterns. In this work, we consider patterns learned with distant supervision and patterns extracted from Universal Schema relations.

The second approach is support vector machines. We evaluate two different feature sets: a bag-of-word feature set (BOW) and more sophisticated skip n-gram features.

Our third approach is a convolutional neural network (CNN). CNNs have been applied to NLP tasks like sentiment analysis, part-of-speech tagging and semantic role labeling. They can recognize phrase patterns independent of their position in the sentence. Furthermore, they make use of word embeddings that directly reflect word similarity (Mikolov et al., 2013). Hence, we expect them to be robust models for the task of classifying filler candidates and to generalize well to unseen test data. In this work, we train different variants of CNNs: As a baseline, we reimplement the recently developed piecewise CNN (Zeng et al., 2015). Then, we extend this model by splitting the contexts not only for

Proceedings of NAACL-HLT 2016, pages 828–838,
San Diego, California, June 12-17, 2016.

pooling but also for convolution (contextwise CNN).

Currently, there is no benchmark for slot filling. Therefore, it is not possible to directly compare results that were submitted to the Shared Task to new results. Comparable manual annotations for new results, for instance, cannot be easily obtained. There are also many different system components, such as document retrieval from the evaluation corpus and coreference resolution, that affect Shared Task performance and that are quite different in nature from relation classification. Even in the subtask of relation classification, it is not possible to directly use existing relation classification benchmarks (e.g. Riedel et al. (2013), Hendrickx et al. (2010)) since data and relations can be quite different. Many benchmark relations, for instance, correspond to Freebase relations but not all slots are modeled in Freebase and some slots even comprise more than one Freebase relation. While most relation classification benchmarks either use newswire or web data, the SF task includes documents from both domains (and discussion fora). Another difference to traditional relation classification benchmarks arises from the pipeline aspect of slot filling. Depending on the previous steps, the input for the relation classification models can be incomplete, noisy, include coreferent mentions, etc.

The official SF Shared Task evaluations only assess whole systems (with potential subsequent faults in their pipelines (Pink et al., 2014)). Thus, we expect component wise comparisons to be a valuable addition to the Shared Task: With comparisons of single components, teams would be able to improve their modules more specifically. To start with one of the most important components, we have created a benchmark for slot filling relation classification, based on 2012 – 2014 TAC Shared Task data. It will be described below and published along with this paper.[1] In addition to presenting model results on this benchmark dataset, we also show that these results correlate with end-to-end SF results. Hence, optimizing a model on this dataset will also help improving results in the end-to-end setting.

In our experiments, we found that our models suffer from large genre differences in the TAC data. Hence, the SF Shared Task is a task that conflates an investigation of domain (or genre) adaptation with the one of slot filling. We argue that both problems are important NLP problems and provide datasets and results for both within and across genres. We hope that this new resource will encourage others to test their models on our dataset and that this will help promote research on slot filling.

In summary, our contributions are as follows. (i) We investigate the complementary strengths and weaknesses of different approaches to relation classification and show that their combination can better deal with a diverse set of problems that slot filling poses than each of the approaches individually. (ii) We propose to split the context at the relation arguments before passing it to the CNN in order to better deal with the special characteristics of a sentence in relation classification. This outperforms the state-of-the-art piecewise CNN. (iii) We analyze the effect of genre on slot filling and show that it is an important conflating variable that needs to be carefully examined in research on slot filling. (iv) We provide a benchmark for slot filling relation classification that will facilitate direct comparisons of models in the future and show that results on this dataset are correlated with end-to-end system results.

Section 2 gives an overview of related work. Section 3 discusses the challenges that slot filling systems face. In Section 4, we describe our slot filling models. Section 5 presents experimental setup and results. Section 6 analyzes the results. We present our conclusions in Section 7 and describe the resources we publish in Section 8.

2 Related Work

Slot filling. The participants of the SF Shared Task (Surdeanu, 2013) are provided with a large text corpus. For evaluation, they get a collection of queries and need to provide fillers for predefined relations and an offset of a context which can serve as a justification. Most participants apply pipeline based systems. Pink et al. (2014) analyzed sources of recall losses in these pipelines. The results of the systems show the difficulty of the task: In the 2014 evaluation, the top-ranked system had an F_1 of .37 (Angeli et al., 2014a). To train their models, most groups use distant supervision (Mintz et al., 2009). The top-ranked systems apply machine learning based ap-

[1] http://cistern.cis.lmu.de

proaches rather than manually developed patterns or models (Surdeanu and Ji, 2014). The methods for extracting and scoring candidates range from pattern based approaches (Gonzàlez et al., 2012; Liu and Zhao, 2012; Li et al., 2012; Qiu et al., 2012; Roth et al., 2014) over rule based systems (Varma et al., 2012) to classifiers (Malon et al., 2012; Roth et al., 2013). The top ranked system from 2013 used SVMs and patterns for evaluating filler candidates (Roth et al., 2013); their results suggest that n-gram based features are sufficient to build reliable classifiers for the relation classification module. They also show that SVMs outperform patterns.

CNNs for relation classification. Zeng et al. (2014) and Dos Santos et al. (2015) apply CNNs to the relation classification SemEval Shared Task data from 2010 and show that CNNs outperform other models. We train CNNs on noisy distant supervised data since (in contrast to the SemEval Shared Task) clean training sets are not available. Malon et al. (2012) describe a CNN for slot filling that is based on the output of a parser. We plan to explore parsing for creating a more linguistically motivated input representation in the future.

Baseline models. In this paper, we will compare our methods against traditional relation classification models: Mintz++ (Mintz et al., 2009; Surdeanu et al., 2012) and MIMLRE (Surdeanu et al., 2012). Mintz++ is a model based on the Mintz features (lexical and syntactic features for relation extraction). It was developed by Surdeanu et al. (2012) and used as a baseline model by them. MIMLRE is a graphical model designed to cope with multiple instances and multiple labels in distant supervised data. It is trained with Expectation Maximization.

Another baseline model which we use in this work is a piecewise convolutional neural network (Zeng et al., 2015). This recently published network is designed especially for the relation classification task which allows to split the context into three parts around the two relation arguments. While it uses the whole context for convolution, it performs max pooling over the three parts individually. In contrast, we propose to split the context even earlier and apply the convolutional filters to each part separately.

Genre dependency. There are many studies showing the genre dependency of machine learning models. In 2012, the SANCL Shared Task focused on evaluating models on web data that have been trained on news data (Petrov and McDonald, 2012). The results show that POS tagging performance can decline a lot when the genre is changed. For other NLP tasks like machine translation or sentiment analysis, this is also a well-known challenge and domain adaptation has been extensively studied (Glorot et al., 2011; Foster and Kuhn, 2007). We do not investigate domain adaptation per se, but show that the genre composition of the slot filling source corpus poses challenges to genre independent models.

3 Challenges of Slot Filling

Slot filling includes NLP challenges of various natures. Given a large evaluation corpus, systems first need to find documents relevant to the entity of the query. This involves challenges like alternate names for the same entity, misspellings of names and ambiguous names (different entities with the same name). Then for each relevant document, sentences with mentions of the entity need to be extracted, as well as possible fillers for the given slot. In most cases, coreference resolution and named entity recognition tools are used for these tasks. Finally, the systems need to decide which filler candidate to output as the solution for the given slot. This step can be reduced to relation classification. It is one of the most crucial parts of the whole pipeline since it directly influences the quality of the final output. The most important challenges for relation classification for slot filling are little or noisy (distant supervised) training data, data from different domains and test sentences which have been extracted with a pipeline of different NLP components. Thus, their quality directly depends on the performance of the whole pipeline. If, for example, sentence splitting fails, the input can be incomplete or too long. If coreference resolution or named entity recognition fails, the relation arguments can be wrong or incomplete.

4 Models for Relation Classification

Patterns. The first approach we evaluate for relation classification is pattern matching. For a given sentence, the pattern matcher classifies the relation as correct if one of the patterns matches; otherwise

the candidate is rejected. In particular, we apply two different pattern sets: The first set consists of patterns learned using distant supervision (*PATdist*). They have been used in the SF challenge by the top-ranked system in the 2013 Shared Task (Roth et al., 2013). The second set contains patterns from universal schema relations for the SF task (*PATuschema*). Universal schema relations are extracted based on matrix factorization (Riedel et al., 2013). In this work, we apply the universal schema patterns extracted for slot filling by Roth et al. (2014).

Support vector machines (SVMs). Our second approach is support vector machines. We evaluate two different feature sets: bag-of-word features (*SVMbow*) and skip n-gram features (*SVMskip*). Based on the results of Roth et al. (2013), we will not use additional syntactic or semantic features for our classifiers. For SVMbow, the representation of a sentence consists of a flag and four bag-of-word vectors. Let m_1 and m_2 be the mentions of name and filler (or filler and name) in the sentence, with m_1 occurring before m_2. The binary flag indicates in which order name and filler occur. The four BOW vectors contain the words in the sentence to the left of m_1, between m_1 and m_2, to the right of m_2 and all words of the sentence. For SVMskip, we use the previously described BOW features and additionally a feature vector which contains skip n-gram features. They wildcard tokens in the middle of the n-gram (cf. Roth et al. (2013)). In particular, we use skip 3-grams, skip 4-grams and skip 5-grams. A possible skip 4-gram of the context ", founder and director of", for example, would be the string "founder of", a pattern that could not have been directly extracted from this context otherwise. We train one linear SVM (Fan et al., 2008) for each relation and feature set and tune parameter C on dev.

Convolutional neural networks (CNNs). CNNs are increasingly applied in NLP (Collobert et al., 2011; Kalchbrenner et al., 2014). They extract n-gram based features independent of the position in the sentence and create (sub-)sentence representations. The two most important aspects that make this possible are convolution and pooling. Max pooling (Collobert et al., 2011) detects the globally most relevant features obtained by local convolution.

Another promising aspect of CNNs for relation classification is that they use an embedding based input representation. With word embeddings, similar words are represented by similar vectors and, thus, we can recognize (near-)synonyms – synonyms of relation triggers as well as of other important context words. If the CNN has learned, for example, that the context "is based in" triggers the relation location_of_headquarters and that "based" has a similar vector representation as "located", it may recognize the context "is located in" correctly as another trigger for the same relation even if it has never seen it during training. In the following paragraphs, we describe the different variants of CNNs which we evaluate in this paper. For each variant, we train one binary CNN per slot and optimize the number of filters ($\in \{300, 1000, 3000\}$), the size of the hidden layer ($\in \{100, 300, 1000\}$) and the filter width ($\in \{3, 5\}$) on dev. We use word2vec (Mikolov et al., 2013) to pre-train word embeddings (dimensionality $d = 50$) on a May-2014 English Wikipedia corpus.

Piecewise CNN. Our baseline CNN is the model developed by Zeng et al. (2015). It represents the input sentence by a matrix of word vectors, applies several filters for convolution and then divides the resulting n-gram representation into left, middle and right context based on the positions m_1 and m_2 of name and filler (see SVM description). For each of the three parts, one max value is extracted by pooling. The results are passed to a softmax classifier.

Contextwise CNN. In contrast to the piecewise CNN, we propose to split the context before convolution as shown in Figure 1. Hence, similar to our BOW vectors for the SVM, we split the original context words into left, middle and right context. Then, we apply convolution and pooling to each of the contexts separately. In contrast to the piecewise CNN, there is no convolution across relation arguments. Thus, the network learns to focus on the context words and cannot be distracted by the presence of (always present) relation arguments. The filter weights W are shared for the three contexts. Our intuition is that the most important sequence features we want to extract by convolution can appear in two or three of the regions. Weight sharing also reduces the number of parameters and increases robustness. We also found in initial experiments that sharing filter weights across left, middle, right outperformed not sharing weights. The results of convolution are pooled using k-max pooling (Kalchbrenner et al.,

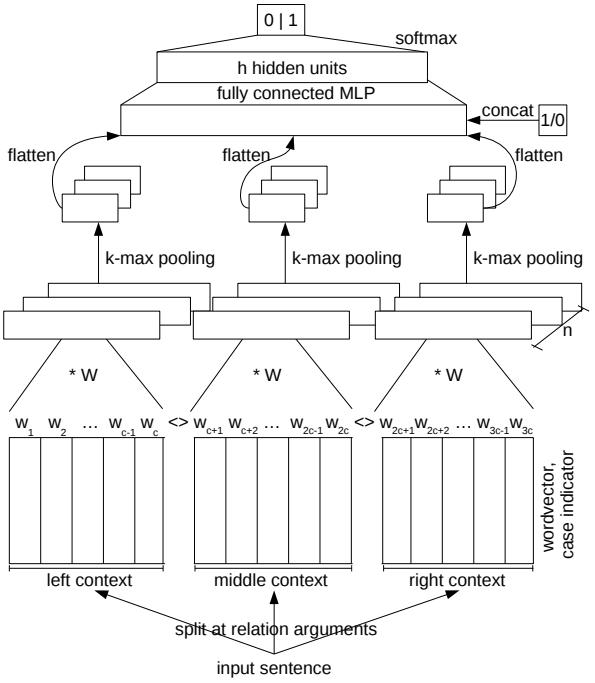

Figure 1: Contextwise CNN for relation classification

2014): only the $k = 3$ maximum values of each filter application are kept. The pooling results are then concatenated to a single vector and extended by a flag indicating whether the name or the filler appeared first in the sentence.

In initial experiments, we found that a fully connected hidden layer after convolution and pooling leads to a more powerful model. It connects the representations of the three contexts and, thus, can draw conclusions based on cooccurring patterns across contexts. Therefore, the result vector after convolution and pooling is fed into a fully connected hidden layer. A softmax layer makes the final decision.

For a fair comparison of models, we also add a hidden layer to the piecewise CNN and apply k-max pooling there as well. Thus, the number of parameters to learn is the same for both models. We call this model *CNNpieceExt*. The key difference between CNNpieceExt and CNNcontext is the time when the context is split into three parts: before or after convolution. This affects the windows of words to which the convolutional filters are applied.

Model combination (CMB). To combine a set M of models for classification, we perform a simple linear combination of the scores of the models:

$$q_{\text{CMB}} = \sum_{m=1...M} \alpha_m q_m$$

where q_m is the score of model m and α_m is its weight (optimized on dev using grid search). All weights sum to 1.

For a comparison of different combination possibilities, see, for example, (Viswanathan et al., 2015).

5 Experiments and Results

5.1 Training Data

We used distant supervision for generating training data. We created a set of (subject, relation, object) tuples by querying Freebase (Bollacker et al., 2008) for relations that correspond to the slot relations. Then we scanned the following corpora for sentences containing both arguments of a relation in the tuple set: (i) the TAC source corpus (TAC, 2014), (ii) a snapshot of Wikipedia (May 2014), (iii) the Freebase description fields, (iv) a subset of Clueweb[2], (v) a New York Times corpus (LDC2008T19). The resulting sentences are positive training examples. Based on the tuple set, we selected negative examples by scanning the corpora for sentences that (i) contain a mention of a name occurring in a tuple, (ii) do not contain the correct filler, (iii) contain a mention different from the correct filler, but with the same named entity type (based on CoreNLP NER (Manning et al., 2014)). All negative examples for date slots, for instance, are sentences containing an incorrect date.

This procedure gave us a large but noisy training set for most slots. In order to reduce incorrect labels, we applied a self-training procedure: We trained SVMs on the SF dataset created by Angeli et al. (2014b). With the resulting SVMs, we predicted labels for our training set. If the predicted label did not match the distant supervised label, we deleted the corresponding training example (Min et al., 2012). This procedure was conducted in several iterations on different chunks of the training set. Finally, the SF dataset and the filtered training examples were merged. (We do not use the SF dataset directly because (i) it provides few examples per slot

[2] http://lemurproject.org/clueweb12

(min: 1, max: 4960) and (ii) it consists of examples for which the classifiers of Angeli et al. (2014b) were indecisive, i.e., presumably contexts that are hard to classify.) Since their contexts are similar, we also merged city, state-or-province and country slots to one location slot.

5.2 Evaluation Data

One of the main challenges in building and evaluating relation classification models for SF is the shortage of training and evaluation data. Each group has their own datasets and comparisons across groups are difficult. Therefore, we have developed a script that creates a clean dataset based on manually annotated system outputs from previous Shared Task evaluations. In the future, it can be used by all participants to evaluate components of their slot filling systems.[3] The script only extracts sentences that contain mentions of both name and filler. It conducts a heuristic check based on NER tags to determine whether the name in the sentence is a valid mention of the query name or is referring to another entity. In the latter case, the example is filtered out. One difficulty is that many published offsets are incorrect. We tried to match these using heuristics. In general, we apply filters that ensure high quality of the resulting evaluation data even if that means that a considerable part of the TAC system output is discarded. In total, we extracted 39,386 high-quality evaluation instances out of the 59,755 system output instances published by TAC and annotated as either completely correct or completely incorrect.

A table in the supplementary material[4] gives statistics: the number of positive and negative examples per slot and year (without duplicates). For 2013, the most examples were extracted. The lower number for 2014 is probably due to the newly introduced inference across documents. This limits the number of sentences with mentions of both name and filler. The average ratio of positive to negative examples is 1:4. The number of positive examples per slot and year ranges from 0 (org:member_of, 2014) to 581 (per:title, 2013), the number of negative examples from 5 (org:website, 2014) to 1886 (per:title, 2013).

In contrast to other relation classification benchmarks, this dataset is not based on a knowledge base (such as Freebase) and unrelated text (such as web documents) but directly on the SF assessments. Thus, it includes exactly the SF relations and addresses the challenges of the end-to-end task: noisy data, possibly incomplete extractions of sentences and data from different domains.

We use the data from 2012/2013 as development and the data from 2014 as evaluation set.

5.3 Experiments

We evaluate the models described in Section 4, select the best models and combine them.

Experiments with patterns. First, we compare the performance of PATdist and PATuschema on our dataset. We evaluate the pattern matchers on all slots presented in Table 1 and calculate their average F_1 scores on dev. PATdist achieves a score of .35, PATuschema of .33. Since it performs better, we use PATdist in the following experiments.

Experiments with SVMs. Second, we train and evaluate SVMbow and SVMskip. Average F_1 of SVMskip and SVMbow are .62 and .59, respectively. Thus, we use SVMskip. We expected that SVMskip beats SVMbow due to its richer feature set, but SVMbow performs surprisingly well.

Experiments with CNNs. Finally, we compare the performance of CNNpiece, CNNpieceExt and CNNcontext. While the baseline network CNNpiece (Zeng et al., 2015) achieves F_1 of .52 on dev, CNNpieceExt has an F_1 score of .55 and CNNcontext an F_1 of .60. The difference of CNNpiece and CNNpieceExt is due to the additional hidden layer and k-max pooling. The considerable difference in performance of CNNpieceExt and CNNcontext shows that splitting the context for convolution has a positive effect on the performance of the network.

Overall results. Table 1 shows the slot wise results of the best patterns (PATdist), SVMs (SVMskip) and CNNs (CNNcontext). Furthermore, it provides a comparison with two baseline models: Mintz++ and MIMLRE. SVM and CNN clearly outperform these baselines. They also outperform PAT for almost all slots. The difference between dev and eval results varies a lot among the slots. We suspect that this is a result of genre differences in the data and analyze this in Section 6.4.

[3] `http://cistern.cis.lmu.de`. We publish scripts since we cannot distribute data.

[4] also available at `http://cistern.cis.lmu.de`

	Mintz++		MIMLRE		PATdist		SVMskip		CNNcontext		CMB	
	dev	eval	dev	eval	dev	eval	dev	eval	dev	eval	dev	eval
per:age	.84	.71	.83	.73	.69	**.80**	**.86**	.74	.83	.76	**.86**	.77
per:alternate_names	.29	.03	.29	.03	**.50**	**.50**	.35	.02	.32	.04	**.50**	**.50**
per:children	.76	.43	.77	.48	.10	.07	.81	.68	.82	.61	**.87**	**.76**
per:cause_of_death	.76	.42	.75	.36	.44	.11	**.82**	.32	.77	**.52**	**.82**	.31
per:date_of_birth	**1.0**	.60	.99	.60	.67	.57	**1.0**	.67	**1.0**	**.77**	**1.0**	.67
per:date_of_death	.67	.45	.67	.45	.30	.32	**.79**	**.54**	.72	.48	**.79**	**.54**
per:empl_memb_of	.38	.36	.41	.37	.24	.22	.42	.36	.41	.37	**.47**	**.39**
per:location_of_birth	.56	.22	.56	.22	.30	.30	.59	.27	.59	.23	**.74**	**.36**
per:loc_of_death	.65	.41	.66	**.43**	.13	.00	.64	.34	.63	.28	**.70**	.35
per:loc_of_residence	.14	.11	.15	.18	.10	.03	**.31**	**.33**	.20	.23	**.31**	.31
per:origin	.40	.48	.42	.46	.13	.11	**.65**	**.64**	.43	.39	**.65**	.59
per:parents	.64	.59	.68	.65	.27	.38	.65	**.79**	.65	.78	**.72**	.71
per:schools_att	.75	**.78**	.76	.75	.27	.26	.78	.71	.72	.55	**.79**	.71
per:siblings	**.66**	.59	.64	.59	.14	.50	.60	.68	.63	**.70**	.65	**.70**
per:spouse	.58	.23	.59	.27	.40	.53	.67	.32	.67	.30	**.78**	**.57**
per:title	.49	.39	.49	.40	.48	.42	.54	**.48**	.57	.46	**.59**	.46
org:alternate_names	.49	.46	.50	.48	.70	**.71**	.62	.62	.65	.66	**.72**	.67
org:date_founded	.41	.71	.42	**.73**	.47	.40	.57	.70	.64	.71	**.68**	.68
org:founded_by	.60	.62	.70	.65	.39	.62	.77	.74	.80	.68	**.85**	**.77**
org:loc_of_headqu	.13	.19	.14	.20	.39	.30	.43	.42	.43	.45	**.50**	**.46**
org:members	.58	.06	.55	.16	.03	**.29**	.70	.13	.65	.04	**.76**	.13
org:parents	.32	.14	.36	.17	.31	.18	.37	.20	.41	.16	**.52**	**.21**
org:subsidiaries	.32	.43	.35	.35	.32	**.56**	.38	.37	.36	.44	**.42**	.49
org:top_memb_empl	.35	.44	.37	.46	.53	.46	.43	**.55**	.43	.53	**.58**	.51
average	.53	.41	.54	.42	.35	.36	.62	.48	.60	.46	**.68**	**.53**

Table 1: Performance on Slot Filling benchmark dataset (dev: data from 2012/2013, eval: from 2014). CMB denotes the combination of PATdist, SVMskip and CNNcontext.

Slot wise results of the other models (PATuschema, SVMbow, CNNpiece, CNNpieceExt) can be found in the supplementary material.

Comparing PAT, SVM and CNN,[5] different patterns emerge for different slots. Each is best on a subset of the slots (see bold numbers). This indicates that relation classification for slot filling is not a uniform problem: each slot has special properties and the three approaches are good at modeling a different subset of these properties. Given the big differences, we expect to gain performance by combining the three approaches. Indeed, CMB (PATdist + SVMskip + CNNcontext), the combination of the three best performing models, obtains the best results in average (in bold).

Section 6.3 shows that the performance on our dataset is highly correlated with SF end-to-end performance. Thus, our results indicate that a combination of different models ist the most promising approach to getting good performance on slot filling.

6 Analysis

6.1 Contribution of Each Model

To see how much each model contributes to CMB, we count how often each weight between 0.0 and 1.0 is selected for the linear interpolation. The results are plotted as a histogram (Figure 2). A weight of 0.0 means that the corresponding model does not contribute to CMB. We see that all three models contribute to CMB for most of the slots. The CNN, for instance, is included in the combination for 14 of 24 slots.

6.2 Comparison of CNN to Traditional Models

Our motivation for using a CNN is that convolution and max pooling can recognize important n-grams independent of their position in the sentence. To in-

[5] In prior experiments, we also compared with recurrent neural networks. RNN performance was comparable to CNNs, but required much more training time and parameter tuning. Therefore, we focus on CNNs in this paper. See also Vu et al. (2016).

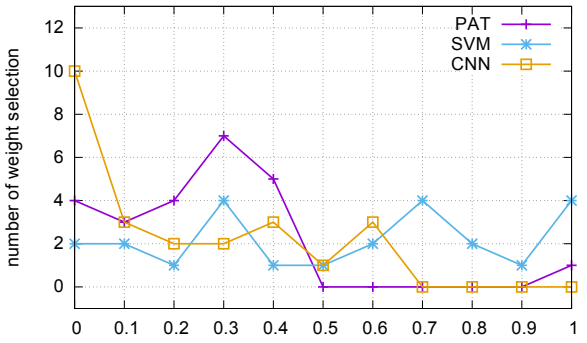

Figure 2: # times each weight is selected in CMB

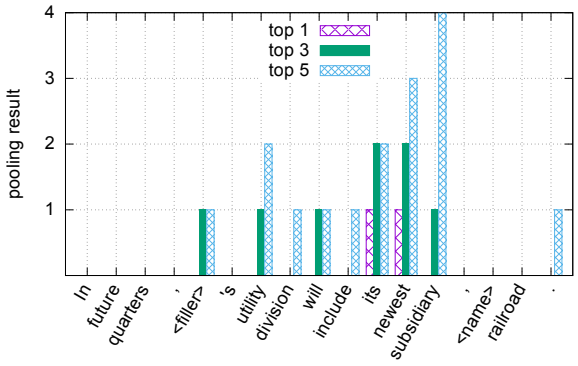

Figure 3: Analysis of convolution and pooling

vestigate this effect, we select for each CNN the top five kernels whose activations are the most correlated with the final score of the positive class. Then we calculate which n-grams are selected by these kernels in the max pooling step. This corresponds to those n-grams which are recognized by the kernel to be the most informative for the given slot. Figure 3 shows the result for an example sentence expressing the slot relation org:parents. The height of a bar is the number of times that the 3-gram around the corresponding word was selected by k-max pooling; e.g., the bar above "newest" corresponds to the trigram "its newest subsidiary". The figure shows that the convolutional filters are able to learn phrases that trigger a relation, e.g., "its subsidiary". In contrast to patterns, they do not rely on exact matches. The first reason is embeddings. They generalize similar words and phrases by assigning similar word vectors to them. For PAT and SVM, this type of generalization is more difficult. The second type of generalization that the CNN learns concerns insertions, similar to skip n-gram features. The recognition of important phrases in convolution is robust against insertions. An example is "newest" in Figure 3, a word that is not important for the slot.

A direct comparison of results with PAT shows that the CNN has better eval scores for about 67% of the slots (see Table 1). Our reasoning above can explain this. Compared to the SVM, the CNN generalizes better to unseen data in only 42% of all cases. The fact that this does not happen in more cases shows the power of the skip n-gram features of the SVM: they also provide a kind of generalization against insertions. The SVM might also need less data to train than the CNN. Nevertheless, the final scores show that the CNN performs almost as well as the SVM in average (.60 vs .62 on dev, .46 vs .48 on eval) and contributes to a better combination score.

6.3 Correlation with End-to-end Results

In this section, we show that using the dataset we provide with this paper allows tuning classification models for the end-to-end SF task. For each model and each possible combination of models, we calculate average results on our evaluation set as well as final F_1 scores when running the whole slot filling pipeline with our in-house system. The best results of our slot filling system are an F_1 of .29 on the 2013 queries and of .25 on the 2014 queries. We calculate Pearson's correlation coefficient to assess correlation of relation classification and end-to-end performances for the n different system configurations (i.e., model combinations). The correlation of the results on our eval dataset with the SF results on 2013 queries is .89, the correlation with the SF results on 2014 queries is .82. This confirms that good results on the dataset we propose lead to good results on the slot filling end-to-end task.

6.4 Effect of Genre and Time

The TAC source corpus consists of about 1M news documents, 1M web documents and 100K documents from discussion forums (TAC, 2014). The distribution of these different genres in the extracted assessment data is as shown in Table 2.

The proportion of non-news more than doubled from 12.5% to 26.6%. Thus, when using 2012/2013 as the development and 2014 as the test set, we are faced with a domain adaptation problem.

	2012/3	2014
news	87.5%	73.4%
web + forums	12.5%	26.6%

Table 2: Distribution of genres

In this section, we show the effect of domain differences on our models in more detail. For our genre analysis, we retrain our models on genre specific training sets WEB and $\text{NEWS}_{\subset}$ and show within-genre as well as cross-genre evaluations. To avoid performance differences due to different training set sizes, we reduced the news training set to the same size as the web training set. We refer to this subset as $\text{NEWS}_{\subset}$.

Cross-genre evaluation. Table 3 shows results of testing models trained on genre-specific data: on data of the same genre and on data of the other genre. We present results only for a subset of relations in this paper, however, the numbers for the other slots follow the same trends.

Models trained on news (left part) show clearly higher performance in the within-genre evaluation than cross-genre. For models trained on web (right part), this is different. We suspect that the reason is that web data is much noiser and thus less predictable, even for models trained on web. For all evaluations, the differences among dev and eval are quite large. Especially for slot filling on web (bottom part of Table 3), the results on dev do not seem much related to the results on eval. This domain effect increases the difficulties of training robust relation classification models for slot filling. It can also explain why optimizing models for unseen data (with unknown genre distributions) as in Table 1 is challenging. Since slot filling by itself is a challenging task, even in the absence of domain differences, we will distribute two splits: a split by year and a split by genre. For training and tuning models for the slot filling research challenge, the year split can be used to cover the challenge of mixing different genres. For experiments on domain adaptation or genre-specific effects, our genre split can be used.

7 Conclusion

In this paper, we presented different approaches to slot filling relation classification: patterns, support vector machines and convolutional neural networks.

		Train on $\text{NEWS}_{\subset}$				Train on WEB			
		SVM		CNN		SVM		CNN	
		dev	ev	dev	ev	dev	ev	dev	ev
Test on news	per:age	.79	.80	**.88**	**.87**	.78	.76	**.85**	**.83**
	per:children	**.85**	**.86**	.78	.78	**.75**	**.80**	.00	.07
	per:spouse	.74	.64	**.76**	**.71**	**.77**	.65	.73	**.67**
	org:alt_names	.22	.32	**.69**	**.67**	.65	**.70**	**.66**	.66
	org:loc_headqu	.51	.50	**.53**	**.51**	.51	**.53**	**.53**	.50
	org:parents	**.30**	.32	.29	**.34**	.26	.33	**.30**	**.34**
Test on web	per:age	.33	.73	**.57**	**.83**	.00	.67	**.57**	**.83**
	per:children	.59	**.33**	**.70**	**.33**	**.63**	**.57**	.00	.00
	per:spouse	.52	.50	**.60**	**.57**	.56	.57	**.67**	**.62**
	org:alt_names	.27	.19	**.51**	**.37**	**.60**	**.49**	.56	.38
	org:loc_headqu	.39	**.46**	**.43**	.44	**.44**	**.48**	.36	.47
	org:parents	.09	**.08**	**.11**	.07	.10	**.08**	**.15**	**.08**

Table 3: Genre specific F_1 scores. Genre specific training data (of the same sizes). Top: news results. Bottom: web results.

We investigated their complementary strengths and weaknesses and showed that their combination can better deal with a diverse set of problems that slot filling poses than each of the approaches individually. We proposed a contextwise CNN which outperforms the recent state-of-the-art piecewise CNN. Furthermore, we analyzed the effect of genre on slot filling and showed that it needs to be carefully examined in research on slot filling. Finally, we provided a benchmark for slot filling relation classification that will facilitate direct comparisons of approaches in the future.

8 Additional Resources

We publish the scripts that we developed to extract the annotated evaluation data and our splits by genre and by year as well as the dev/eval splits.

Acknowledgments

Heike Adel is a recipient of the Google European Doctoral Fellowship in Natural Language Processing and this research is supported by this fellowship.

This research was also supported by Deutsche Forschungsgemeinschaft: grant SCHU 2246/4-2.

We would like to thank Gabor Angeli for his help with the Mintz++ and MIMLRE models.

References

Gabor Angeli, Sonal Gupta, Melvin Jose, Christopher D. Manning, Christopher Re, Julie Tibshirani, Jean Y. Wu, Sen Wu, and Ce Zhang. 2014a. Stanfords 2014 slot filling systems. In *TAC*.

Gabor Angeli, Julie Tibshirani, Jean Y. Wu, and Christopher D. Manning. 2014b. Combining distant and partial supervision for relation extraction. In *EMNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*.

Cícero Nogueira Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *ACL*.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *JMLR*.

George Foster and Roland Kuhn. 2007. Mixture-model adaptation for SMT. In *Workshop on SMT*.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*.

E. Gonzàlez, H. Rodríguez, J. Turmo, P. R. Comas, A. Naderi, A. Ageno, E. Sapena, M. Vila, and M. A. Martí. 2012. The TALP participation at TAC-KBP 2012. In *TAC*.

Iris Hendrickx, Su Nam Kim, Zornitsa Kozareva, Preslav Nakov, Diarmuid Ó Séaghdha, Sebastian Padó, Marco Pennacchiotti, Lorenza Romano, and Stan Szpakowicz. 2010. Semeval-2010 task 8: Multi-way classification of semantic relations between pairs of nominals. In *SemEval*. ACL.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. In *ACL*.

Yan Li, Sijia Chen, Zhihua Zhou, Jie Yin, Hao Luo, Liyin Hong, Weiran Xu, Guang Chen, and Guo Jun. 2012. PRIS at TAC 2012 KBP track. In *TAC*.

Fang Liu and Jun Zhao. 2012. Sweat2012: Pattern based English slot filling system for knowledge base population at TAC 2012. In *TAC*.

Christopher Malon, Bing Bai, and Kazi Saidul Hasan. 2012. Slot-filling by substring extraction at TAC KBP 2012 (team papelo). In *TAC*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL: System Demonstrations*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *Workshop at ICLR*.

Bonan Min, Xiang Li, Ralph Grishman, and Ang Sun. 2012. New york university 2012 system for KBP slot filling. In *TAC*.

Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-IJCNLP*.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *SANCL*.

Glen Pink, Joel Nothman, and James R Curran. 2014. Analysing recall loss in named entity slot filling. In *EMNLP*.

Xin Ying Qiu, Xiaoting Li, Weijian Mo, Manli Zheng, and Zhuhe Zheng. 2012. GDUFS at slot filling TAC-KBP 2012. In *TAC*.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *HLT-NAACL*.

Benjamin Roth, Tassilo Barth, Michael Wiegand, Mittul Singh, and Dietrich Klakow. 2013. Effective slot filling based on shallow distant supervision methods. In *TAC*.

Benjamin Roth, Emma Strubell, John Sullivan, Lakshmi Vikraman, Kate Silverstein, and Andrew McCallum. 2014. Universal schema for slot-filling, cold-start KBP and event argument extraction: UMAss IESL at TAC KBP 2014. In *TAC*.

Mihai Surdeanu and Heng Ji. 2014. Overview of the English slot filling track at the TAC 2014 knowledge base population evaluation. In *TAC*.

Mihai Surdeanu, Julie Tibshirani, Ramesh Nallapati, and Christopher D Manning. 2012. Multi-instance multi-label learning for relation extraction. In *EMNLP-CoNLL*.

Mihai Surdeanu. 2013. Overview of the TAC 2013 knowledge base population evaluation: English slot filling and temporal slot filling. In *TAC*.

TAC. 2014. Task description for English slot filling at TAC KBP 2014. `http://surdeanu.info/kbp2014/KBP2014_TaskDefinition_EnglishSlotFilling_1.1.pdf`.

Vasudeva Varma, Bhaskar Ghosh, Mohan Soundararajan, Deepti Aggarwal, and Priya Radhakrishnan. 2012. IIIT Hyderabad at TAC 2012. In *TAC*.

Vidhoon Viswanathan, Nazneen Fatema Rajani, Yinon Bentor, and Raymond Mooney. 2015. Stacked ensembles of information extractors for knowledge-base population. In *ACL*.

Ngoc Thang Vu, Heike Adel, Pankaj Gupta, and Hinrich Schütze. 2016. Combining recurrent and convolutional neural networks for relation classification. In *HLT-NAACL*.

Daojian Zeng, Kang Liu, Siwei Lai, Guangyou Zhou, and Jun Zhao. 2014. Relation classification via convolutional deep neural network. In *COLING*.

Daojian Zeng, Kang Liu, Yubo Chen, and Jun Zhao. 2015. Distant supervision for relation extraction via piecewise convolutional neural networks. In *EMNLP*.