Domain Adaptation for Relation Extraction

Daniel Posch, Sheldon Chang, Matthew Watson
{dcposch, sheldonc, mdwatson}@stanford.edu
Mentored by Mihai Surdeanu and David McClosky

2010.12.10

Abstract

We describe our experiments with domain adaptation strategies in order to boost the performance of the JavaNLP relation extraction system built by Stanford's natural language processing group. We implemented a domain adaptation preprocessing step known as EasyAdapt as well as a confidence-weighted linear classifier with domain adaptation and compared them to several baseline strategies[6]. EasyAdapt outperformed all baselines.

1 Introduction

"Relation extraction" means identifying instances of relations (such as 'created-by' or 'part-of'), in our case from a body of text and a set of initial examples. It is part of the larger field of knowledge base population. Our goal was to make an existing relation extractor perform well on text from multiple domains, such as Wikipedia and newsgroup articles. The system begins with positive examples of each relation supplied–in other words, the knowledge base does not start out empty.

The main focus of our project is applying domain adaptation strategies to the KBP system in the hope of improved performance. Traditionally, domain adaption has been about making learning models perform well on domains other than the training, or "source" domain. In our case, both our training and testing data is heterogenous (ie, from multiple domains). We apply domain adaptation with the intuition that different sources of training data will provide different feature weights, so it is a mistake to treat them all equally.

To our knowledge, and as far as our mentors in the NLP group are aware, we are the first to explore domain adaptation on a complex information extraction system.

2 Prior Work

2.1 Relation Extraction

Several different approaches have been tried for algorithmically extracting relations from textual data. Such algorithms might discover, for example, that Ronald Reagan succeeded Jimmy Carter in office, or that wheels are part of a car. They generally fall into supervised approaches, which take a corpus of text labeled with known entities and relations, and unsupervised approaches, which discover new relations directly from unlabeled text. The Stanford KBP algorithm[2] focuses on an in-between approach described in [3, Mintz et al] – relation extraction via distant supervision. The algorithm parses input text into a collection of sentences. An example for a relation is simply a pair of entities, and each sentence containing that pair is assumed to express that relation. If one of the examples for `person:child` is `<Darth Vader, Luke Skywalker>`, then the sentence "In Episode VI, Luke Skywalker is shocked to learn that Darth Vader is his father" would match. Each tuple of `<entity1, entity2, sentence>` is then turned into a large vector of features–for example, the words that occur between the two entities and the relative positions of the entities in a syntactic parse tree. Each feature is encoded as a string.

Finally, the algorithm creates a set of training examples $\{x_i, y_i\}$ where x_i are feature vectors and y_i are the corresponding relation labels, known in Knowledge Base Population as "slots". It trains a classifier

on this data. During testing, the system is queried with an entity and asked to populate the predefined relations using other named entities that appear in sentences with the query entity. These sentences, like the training examples, are drawn from newsgroup text, web snippets, and Wikipedia. A candidate feature vector is then constructed for the original entity and the extracted named entity, and this feature vector is then classified as one of the relations or a "no relation" category.

2.2 Domain Adaptation

Domain adaptation in NLP is the challenge of enabling algorithms to perform well on text written in the same language but in different formats or styles, and to be able to train algorithms on similarly disparate input.

We have experimented with several domain adaptation strategies.

3 Methods

For this project, we sought to improve the performance of the existing KBP model by implementing the EasyAdapt feature augmentation strategy (Daume 2007), and the confidence weighted multi-domain linear classifier described in [6, Dredze and Crammer].

We evaluated these systems against several baselines:

ALL using the union of all three domains as the source domain for the classifier (this is the behavior of the original KBP system implemented by Surdeanu et al)

WEBONLY, NEWSONLY, WIKIONLY using each domain individually as a source domain (e.g. training on just web snippets, then testing on all three domains)

LININT training three separate classifiers on each domain, then linearly interpolating the resulting classification probabilities

As Daume noted, these baselines are surprisingly difficult to beat. His EasyAdapt strategy managed to surpass all of these baselines (and several others) on a large variety of datasets, making it one of the few models for domain adaption to have done so.

We evaluated the baselines and the domain-adapted systems on a smaller version of the KBP knowledge base (20% of the entire data available), with half of the data used as training and the other half as testing. The train and test sets each contained roughly 300MB of text. Sentences were extracted with a known slot type in the same manner as in training. We then scored the classifiers' ability to predict the correct slot type for a given sentence by measuring precision (p), recall (r), and F-score ($F_1 = \frac{p+r}{2pr}$).

3.1 EasyAdapt

The first and simplest strategy we implemented is described in [4, Daume 2007]. In that approach, we simply expand the feature space by adding one copy of each feature per domain, in addition to the generic, domain-independent original. For example, if a sentence comes from a web snippet and has feature `x`, then we will augment its feature vector to also contain `web-x`. It will not contain `news-x` or `wiki-x`. If the original feature space was n-dimensional and we are adapting our algorithm to d domains, then the new feature space is $n(d+1)$-dimensional – each feature now has a general version as well as d domain-specific ones. When we train a classifier, we are determining the relative importance of each feature for predicting instances of each relation, as well as the importance of each feature in each specific domain.

3.2 Dredze and Crammer

We implemented a more involved domain adaptation strategy based on a generalization of the perceptron algorithm. Confidence-weighted online linear classifiers work by assigning a variance to each feature weight [5]. Whenever a new example x_i arrives, the classifier updates its weights as little as possible to accomodate it. Following the authors' convention, the weights after step i are μ_i and the variances Σ_i, represented as a diagonal covariance matrix. The update rules are:

$$\mu_{i+1} = \mu_i + \alpha y_i \Sigma_i x_i$$
$$\Sigma_{i+1}^{-1} = \Sigma_i^{-1} + 2\alpha\phi x_i x_i^T$$

As Dredze notes, the uncertainties associated with each weight are strictly decreasing as training progresses.

For our setting, we needed the classifier to handle multiple labels; the classifier described in the paper, however, is binary. We used another technique, by Crammer, to do multiclass classification [7]. Unlike its binary counterpart, the multiclass CW classifier

has an update rule that cannot be expressed as convex optimization, nor is it solvable in closed form. As a result, we had to choose among the several heuristic update functions the paper described. We chose the simplest function to start with–the single-constraint update. This takes a form very similar to the update rule of the binary classifier. In our case, it simplifies to

$$\mu_{i+1,y_i} = \mu_i + \alpha_i \Sigma_{i,y_i} x_i$$

$$\mu_{i+1,r} = \mu_i - \alpha_i \Sigma_{i,r} x_i$$

$$\Sigma i+1, l = \left(\Sigma_{i,l}^{-} 1 + 2\alpha_i \phi 1\{l = y_i \cup l = r\} x_i x_i^T\right)^{-1}$$

...where a_i is given by

$$\frac{-(1-2\phi m_i) + \sqrt{(1+2\phi m_i)^2 - 8\phi(m_i - \phi v_i)}}{4\phi v_i}$$

$m_i = \mu_{i,y_i} \cdot x_i - \mu_{i,r} \cdot x_i$ and ϕ is a learning aggressivness parameter. Here, y_i is the correct label for x_i, while r is the highest-scoring label accoring to the weights after step i. Much like a perceptron, no change is made to the parameters when the system guesses correctly, ie when $y_i = r$.

We implemented and tested this classifier by itself and using Daume's feature augmentation as a preprocessing step.

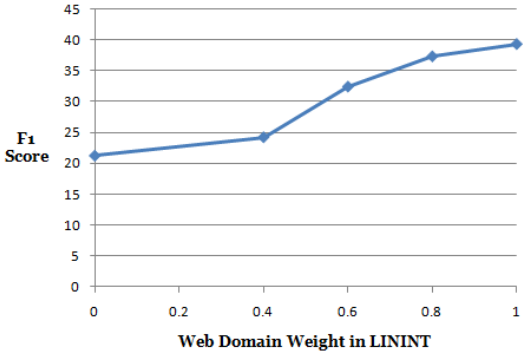
Finally, we implemented domain adaptation by leveraging the measures of confidence the classifier is able to assign to each feature weight. We trained a confidence-weighted classifier separately for each source domain, then linearly combined the resulting weight vectors according to each weight's confidence, as described by [6, Dredze and Crammer]. That paper described four different ways to combine weights. We picked one that was both simple, and, in their tests, effective. It simply takes the weighted average of the classifier μ, weighting each element by a constant minus the corresponding sigma–so that elements with low sigma (high certainty) are weighted heavily.

4 Results

We had a few surprises come out of our results. EasyAdapt was the only domain adaptation strategy that produced an improvement in F-score over the original algorithm (ALL, which simply lumps text from multiple domains together as one input corpus). Interestingly, text from web snippets turned out to be almost as good as the entire input corpus; we had initially suspected that this would be the case for the more consistent WIKI sentences.

LININT, which interpolates between WEB, WIKI, and NEWS, achieved its highest score by only using WEB, as shown below:

	Precision	Recall	F1
ALL	58.4	30.3	39.9
WEBONLY	52.5	31.4	39.3
NEWSONLY	49.1	6.2	10.9
WIKIONLY	58.8	17.2	26.7
LININT (equal weights)	70.0	11.8	20.2
CW Multiclass Linear	36.9	35.2	36
CW Multiclass Linear w/ EA	39.2	37.3	38.2
Dredze & Crammer	36.0	33.4	34.6
EasyAdapt (EA)	57.3	32.2	41.3



Finally, we evaluated the original, unmodified JavaNLP relation extractor on the same labels (slots) shown in [3, Mintz et al]:

Label	Correct	Predict	Actual	Precn	Recall	F1
_NR	633724	687450	649502	92.2	97.6	94.8
org:city_of_headquarters	2639	4229	6877	62.4	38.4	47.5
org:country_of_headquarters	830	1461	2907	56.8	28.6	38
org:founded	1493	2369	3972	63	37.6	47.1
org:parents	298	543	2150	54.9	13.9	22.1
org:top_membersSLASHemployees	1722	3328	9876	51.7	17.4	26.1
per:city_of_birth	832	1781	3158	46.7	26.3	33.7
per:country_of_birth	583	1354	2824	43.1	20.6	27.9
per:date_of_birth	8920	11968	12376	74.5	72.1	73.3
per:member_of	208	376	1378	55.3	15.1	23.7
per:title	2236	5502	11912	40.6	18.8	25.7
...						
Total	23897	40946	78894	**58.4**	**30.3**	**39.9**

Using EasyAdapt, we achieved improvement over almost all relation slots. Considering that JavaNLP is fairly mature software, we are pleased with this result.

Label	Correct	Predict	Actual	Precn	Recall	F1
_NR	631933	683961	649502	92.4	97.3	94.8
org:city_of_headquarters	2655	4295	6877	61.8	38.6	47.5
org:country_of_headquarters	785	1436	2907	54.7	27	36.2
org:founded	1573	2532	3965	62.1	39.7	48.4
org:parents	342	641	2153	53.4	15.9	24.5
org:top_membersSLASHemployees	2341	4398	9916	53.2	23.6	32.7
per:city_of_birth	841	1813	3174	46.4	26.5	33.7
per:country_of_birth	636	1511	2839	42.1	22.4	29.2
per:date_of_birth	9004	12089	12384	74.5	72.7	73.6
per:member_of	180	398	1327	45.2	13.6	20.9
per:title	2585	6562	11912	39.4	21.7	28
...						
Total	25470	44435	78894	**57.3**	**32.3**	**41.3**

5 Future Work

We may explore additional domain adaptation strategies as the NLP group prepares for the 2010 Text Analysis Conference. We are hoping to enter submit our results to a workshop at this conference. Stanford's relation extraction system may also compete in the conference's Knowledge Base Population competition. With Mihai's guidance, we are considering several future directions for the project:

- Switching our classifier from mention-level to relation-level. In other words, collapse all instances of a given relation, e.g., (Bill Gates, org:founder, Microsoft) into a single vector. Based on [8, Riedel et al., ECML 2010].
- Testing the remaining three classifier weight combination methods described in [Dredze, Crammer].
- Implementing a more sophisticated update heuristic for the online multiclass confidence-weighted classifier described in [Dredze et al]. The paper notes that the cost function each update tries to minimize is not convex, nor does it have a closed-form solution; as such, updates are necessarily heuristic.

6 Discussion and conclusion

As our results indicate, for most relations the EasyAdapt classifier outperformed the baselines. We suspect that the reason why we saw these improvements is because augmenting the feature space allows the system to account for the fact that domains that see a lot of a certain type of relation will perform better on that relation. Contrast this with LININT, where even if a certain relation was predicted with high confidence in one domain, it was often negated by the fact that the other domains had not seen the relation and predicted a NR label.

The Dredze and Crammer model is theoretically immune to this problem, since it takes confidence into account. However, it did not perform quite as well as EasyAdapt. This may be because the Drezde's multiclass confidence weighted linear classifier has an imprecise (heuristic) update step. We may experiment with different update steps and different weighting algorithms in future work.

LININT tended to push every predicted label towards the NR (no relation) label. It would only predict a positive label when there was a fair amount of confidence across all domains, which did not happen often. This lead to high accuracy as it was making obvious choices, but poor overall results.

In our examination of the feature weights across all models, it was clear that some features were highly predictive for certain categories. For example, the feature `span_word:chairman` had a very high weight for the labeling `organization:top_membersSLASHemployees`. Other features that consistently demonstrated high feature weights include the "trigger words" used by Surdeanu et al in their sentence retrieval. However, other features had very little predictive power, and from initial observations, the parse tree features did not carry a lot of weight. Experimenting with feature categories and seeing which ones the classifier deems important is one area we can investigate in the future, as using fewer but more relevant features would improve training speed. Preliminary experiments in feature tweaking were able to further boost our F1 score by roughly 0.8.

References

[1] *Task Description for Knowledge Base Population at TAC 2010* `http://nlp.cs.qc.cuny.edu/kbp/2010/KBP2010_TaskDefinition.pdf`

[2] Mihai Surdeanu, et al. *A Simple Distant Supervision Approach for the TAC-KBP Slot Filling Task* Proceedings of the TAC-KBP Workshop, 2010.

[3] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. 2008. *Distant Supervision for Relation Extraction without labeled data* `http://www.stanford.edu/ jurafsky/mintz.pdf`

[4] Hal Daume III. 2007. *Frustratingly Easy Domain Adaptation* `http://www.umiacs.umd.edu/ hal/docs/daume07easyadap`

[5] Mark Dredze et al. 2007. *Confidence-Weighted Linear Classification* `http://www.cs.jhu.edu/ mdredze/publications/icml_variance.pdf`

[6] Mark Dredze and Koby Crammer. 2008. *Online Methods for Multi-Domain Learning and Adaptation* `http://www.cs.jhu.edu/ mdredze/publications/multi_domain_emnlp08.pdf`

[7] Koby Crammer et al. 2009. *Multi-Class Confidence Weighted Algorithms* `http://www.aclweb.org/anthology/D/D09/D09-1052.pdf`

[8] Sebastian Riedel et al. ECML 2010. *Modeling Relations and Their Mentions without Labeled Text.* `http://www.springerlink.com/content/3457744035qm6rw5/fulltext.pdf`