Unsupervised Discovery of Domain-Specific Knowledge from Text

Dirk Hovy, Chunliang Zhang, Eduard Hovy
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Marina del Rey, CA 90292
{dirkh, czheng, hovy}@isi.edu

Anselmo Peñas
UNED NLP and IR Group
Juan del Rosal 16
28040 Madrid, Spain
anselmo@lsi.uned.es

Abstract

Learning by Reading (LbR) aims at enabling machines to acquire knowledge from and reason about textual input. This requires knowledge about the domain structure (such as entities, classes, and actions) in order to do inference. We present a method to infer this implicit knowledge from unlabeled text. Unlike previous approaches, we use automatically extracted classes with a probability distribution over entities to allow for context-sensitive labeling. From a corpus of 1.4m sentences, we learn about 250k simple propositions about American football in the form of predicate-argument structures like "quarterbacks throw passes to receivers". Using several statistical measures, we show that our model is able to generalize and explain the data statistically significantly better than various baseline approaches. Human subjects judged up to 96.6% of the resulting propositions to be sensible. The classes and probabilistic model can be used in textual enrichment to improve the performance of LbR end-to-end systems.

1 Introduction

The goal of Learning by Reading (LbR) is to enable a computer to learn about a new domain and then to reason about it in order to perform such tasks as question answering, threat assessment, and explanation (Strassel et al., 2010). This requires joint efforts from Information Extraction, Knowledge Representation, and logical inference. All these steps depend on the system having access to basic, often unstated, foundational knowledge about the domain.

Most documents, however, do not explicitly mention this information in the text, but assume basic background knowledge about the domain, such as positions ("quarterback"), titles ("winner"), or actions ("throw") for sports game reports. Without this knowledge, the text will not make sense to the reader, despite being well-formed English. Luckily, the information is often implicitly contained in the document or can be inferred from similar texts.

Our system automatically acquires domain-specific knowledge (classes and actions) from large amounts of unlabeled data, and trains a probabilistic model to determine and apply the most informative classes (*quarterback*, etc.) at appropriate levels of generality for unseen data. E.g., from sentences such as "Steve Young threw a pass to Michael Holt", "Quarterback Steve Young finished strong", and "Michael Holt, the receiver, left early" we can learn the classes *quarterback* and *receiver*, and the proposition "*quarterbacks throw passes to receivers*".

We will thus assume that the implicit knowledge comes in two forms: actions in the form of predicate-argument structures, and classes as part of the source data. Our task is to identify and extract both. Since LbR systems must quickly adapt and scale well to new domains, we need to be able to work with large amounts of data and minimal supervision. Our approach produces simple propositions about the domain (see Figure 1 for examples of actual propositions learned by our system).

American football was the first official evaluation domain in the DARPA-sponsored Machine Reading program, and provides the background for a number

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 1466–1475,
Portland, Oregon, June 19-24, 2011.

of LbR systems (Mulkar-Mehta et al., 2010). Sports is particularly amenable, since it usually follows a finite, explicit set of rules. Due to its popularity, results are easy to evaluate with lay subjects, and game reports, databases, etc. provide a large amount of data. The same need for basic knowledge appears in all domains, though. In music, *musicians* play *instruments*, in electronics, *components* constitute *circuits*, *circuits* use *electricity*, etc.

Teams beat teams
Teams play teams
Quarterbacks throw passes
Teams win games
Teams defeat teams
Receivers catch passes
Quarterbacks complete passes
Quarterbacks throw passes to receivers
Teams play games
Teams lose games

Figure 1: The ten most frequent propositions discovered by our system for the American football domain

Our approach differs from verb-argument identification or Named Entity (NE) tagging in several respects. While previous work on verb-argument selection (Pardo et al., 2006; Fan et al., 2010) uses fixed sets of classes, we cannot know a priori how many and which classes we will encounter. We therefore provide a way to derive the appropriate classes automatically and include a probability distribution for each of them. Our approach is thus less restricted and can learn context-dependent, fine-grained, domain-specific propositions. While a NE-tagged corpus could produce a general proposition like "*PERSON* throws to *PERSON*", our method enables us to distinguish the arguments and learn "*quarterback* throws to *receiver*" for American football and "*outfielder* throws to *third base*" for baseball. While in NE tagging each word has only one correct tag in a given context, we have hierarchical classes: an entity can be correctly labeled as a *player* or a *quarterback* (and possibly many more classes), depending on the context. By taking context into account, we are also able to label each sentence individually and account for unseen entities without using external resources.

Our contributions are:

- we use unsupervised learning to train a model that makes use of automatically extracted classes to uncover implicit knowledge in the form of predicate-argument propositions
- we evaluate the explanatory power, generalization capability, and sensibility of the propositions using both statistical measures and human judges, and compare them to several baselines
- we provide a model and a set of propositions that can be used to improve the performance of end-to-end LbR systems via textual enrichment.

2 Methods

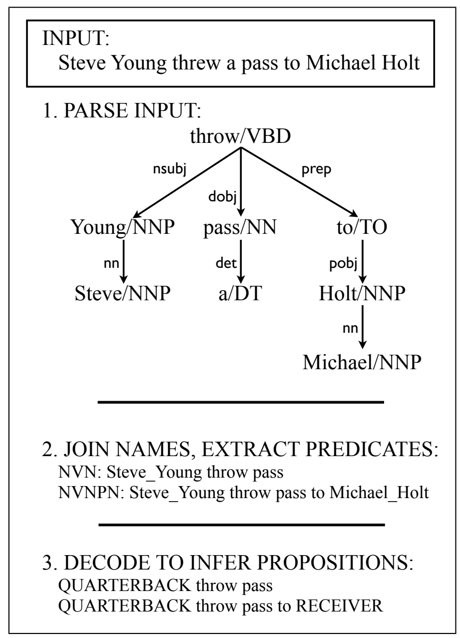

Figure 2: Illustrated example of different processing steps

Our running example will be "Steve Young threw a pass to Michael Holt". This is an instance of the underlying proposition "*quarterbacks throw passes to receivers*", which is not explicitly stated in the data. A proposition is thus a more general statement about the domain than the sentences it derives. It contains domain-specific classes (*quarterback*, *receiver*), as well as lexical items ("throw", "pass"). In order to reproduce the proposition, given the input sentences, our system has to not only identify the classes, but also learn when to abstract away from the lexical form to the appropriate class and when to keep it (cf. Figure 2, step 3). To facilitate extraction, we focus on propositions with the following predicate-argument structures: NOUN-VERB-NOUN (e.g., "*quarterbacks throw passes*"), or NOUN-VERB-NOUN-PREPOSITION-NOUN (e.g., "*quarterbacks throw passes to receivers*". There is nothing, though, that prevents the use of other types of structures as well. We do not restrict the verbs we consider (Pardo et al., 2006; Ritter et al., 2010)), which extracts a high number of hapax structures.

Given a sentence, we want to find the most likely class for each word and thereby derive the most likely proposition. Similar to Pardo et al. (2006), we assume the observed data was produced by a process that generates the proposition and then transforms the classes into a sentence, possibly adding additional words. We model this as a Hidden Markov Model (HMM) with bigram transitions (see Section 2.3) and use the EM algorithm (Dempster et al., 1977) to train it on the observed data, with smoothing to prevent overfitting.

2.1 Data

We use a corpus of about 33k texts on American football, extracted from the New York Times (Sandhaus, 2008). To identify the articles, we rely on the provided "football" keyword classifier. The resulting corpus comprises $1,359,709$ sentences from game reports, background stories, and opinion pieces. In a first step, we parse all documents with the Stanford dependency parser (De Marneffe et al., 2006) (see Figure 2, step 1). The output is lemmatized (collapsing "*throws*", "*threw*", etc., into "*throw*"), and marked for various dependencies (*nsubj*, *amod*, etc.). This enables us to extract the predicate argument structure, like subject-verb-object, or additional prepositional phrases (see Figure 2, step 2). These structures help to simplify the model by discarding additional words like modifiers, determiners, etc., which are not essential to the proposition. The same approach is used by (Brody, 2007). We also concatenate multi-word names (identified by sequences of NNPs) with an underscore to form a single token ("Steve/NNP Young/NNP" → "Steve_Young").

2.2 Deriving Classes

To derive the classes used for entities, we do not restrict ourselves to a fixed set, but derive a domain-specific set directly from the data. This step is performed simultaneously with the corpus generation described above. We utilize three syntactic constructions to identify classes, namely *nominal modifiers*, *copula verbs*, and *appositions*, see below. This is similar in nature to Hearst's lexico-syntactic patterns (Hearst, 1992) and other approaches that derive *IS-A* relations from text. While we find it straightforward to collect classes for entities in this way, we did not find similar patterns for verbs. Given a suitable mechanism, however, these could be incorporated into our framework as well.

Nominal modifier are common nouns (labeled NN) that precede proper nouns (labeled NNP), as in "***quarterback/NN** Steve/NNP Young/NNP*", where "quarterback" is the nominal modifier of "Steve Young". Similar information can be gained from appositions (e.g., "*Steve Young, **the quarterback of his team**, said...*"), and copula verbs ("*Steve Young **is the quarterback** of the 49ers*"). We extract those co-occurrences and store the proper nouns as entities and the common nouns as their possible classes. For each pair of class and entity, we collect counts over the corpus to derive probability distributions.

Entities for which we do not find any of the above patterns in our corpus are marked "*UNK*". These entities are instantiated with the 20 most frequent classes. All other (non-entity) words (including verbs) have only their identity as class (i.e., "*pass*" remains "*pass*").

The average number of classes per entity is 6.87. The total number of distinct classes for entities is $63,942$. This is a huge number to model in our state space.[1] Instead of manually choosing a subset of the classes we extracted, we defer the task of finding the best set to the model.

We note, however, that the distribution of classes for each entity is highly skewed. Due to the unsupervised nature of the extraction process, many of the extracted classes are hapaxes and/or random noise. Most entities have only a small number of applicable classes (a football player usually has one main posi-

[1]NE taggers usually use a set of only a few dozen classes at most.

tion, and a few additional roles, such as *star*, *teammate*, etc.). We handle this by limiting the number of classes considered to 3 per entity. This constraint reduces the total number of distinct classes to $26,165$, and the average number of classes per entity to 2.53. The reduction makes for a more tractable model size without losing too much information. The class alphabet is still several magnitudes larger than that for NE or POS tagging. Alternatively, one could use external resources such as Wikipedia, Yago (Suchanek et al., 2007), or WordNet++ (Ponzetto and Navigli, 2010) to select the most appropriate classes for each entity. This is likely to have a positive effect on the quality of the applicable classes and merits further research. Here, we focus on the possibilities of a self-contained system without recurrence to outside resources.

The number of classes we consider for each entity also influences the number of possible propositions: if we consider exactly one class per entity, there will be little overlap between sentences, and thus no generalization possible—the model will produce many distinct propositions. If, on the other hand, we used only one class for all entities, there will be similarities between many sentences—the model will produce very few distinct propositions.

2.3 Probabilistic Model

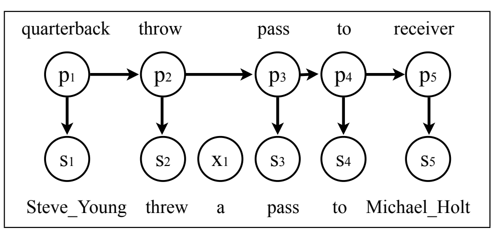

Figure 3: Graphical model for the running example

We use a generative noisy-channel model to capture the joint probability of input sentences and their underlying proposition. Our generative story of how a sentence $\mathbf{s}$ (with words $s_1, ..., s_n$) was generated assumes that a proposition $\mathbf{p}$ is generated as a sequence of classes $p_1, ..., p_n$, with transition probabilities $P(p_i|p_{i-1})$. Each class p_i generates a word s_i with probability $P(s_i|p_i)$. We allow additional words x in the sentence which do not depend on any class in the proposition and are thus generated independently with $P(x)$ (cf. model in Figure 3).

Since we observe the co-occurrence counts of classes and entities in the data, we can fix the emission parameter $P(s|p)$ in our HMM. Further, we do not want to generate sentences from propositions, so we can omit the step that adds the additional words x in our model. The removal of these words is reflected by the preprocessing step that extracts the structure (cf. Section 2.1).

Our model is thus defined as

$$P(\mathbf{s}, \mathbf{p}) = P(p_1) \cdot \prod_{i=1}^{n} \Big(P(p_i|p_{i-1}) \cdot P(s_i|p_i) \Big) \quad (1)$$

where s_i, p_i denote the i^{th} word of sentence $\mathbf{s}$ and proposition $\mathbf{p}$, respectively.

3 Evaluation

We want to evaluate how well our model predicts the data, and how sensible the resulting propositions are. We define a good model as one that generalizes well and produces semantically useful propositions.

We encounter two problems. First, since we derive the classes in a data-driven way, we have no gold standard data available for comparison. Second, there is no accepted evaluation measure for this kind of task. Ultimately, we would like to evaluate our model externally, such as measuring its impact on performance of a LbR system. In the absence thereof, we resort to several complementary measures, as well as performing an annotation task. We derive evaluation criteria as follows. A model generalizes well if it can cover ('explain') all the sentences in the corpus with a few propositions. This requires a measure of generality. However, while a proposition such as "*PERSON* does *THING*", has excellent generality, it possesses no discriminating power. We also need the propositions to partition the sentences into clusters of semantic similarity, to support effective inference. This requires a measure of distribution. Maximal distribution, achieved by assigning every sentence to a different proposition, however, is not useful either. We need to find an appropriate level of generality within which the sentences are clustered into propositions for the best overall groupings to support inference.

To assess the learned model, we apply the measures of *generalization*, *entropy*, and *perplexity* (see

Sections 3.2, 3.3, and 3.4). These measures can be used to compare different systems. We do not attempt to weight or combine the different measures, but present each in its own right.

Further, to assess label accuracy, we use Amazon's Mechanical Turk annotators to judge the sensibility of the propositions produced by each system (Section 3.5). We reason that if our system learned to infer the correct classes, then the resulting propositions should constitute true, general statements about that domain, and thus be judged as sensible.[2] This approach allows the effective annotation of sufficient amounts of data for an evaluation (first described for NLP in (Snow et al., 2008)).

3.1 Evaluation Data

With the trained model, we use Viterbi decoding to extract the best class sequence for each example in the data. This translates the original corpus sentences into propositions. See steps 2 and 3 in Figure 2.

We create two baseline systems from the same corpus, one which uses the most frequent class (MFC) for each entity, and another one which uses a class picked at random from the applicable classes of each entity.

Ultimately, we are interested in labeling unseen data from the same domain with the correct class, so we evaluate separately on the full corpus and the subset of sentences that contain unknown entities (i.e., entities for which no class information was available in the corpus, cf. Section 2.2).

For the latter case, we select all examples containing at least one unknown entity (labeled UNK), resulting in a subset of $41,897$ sentences, and repeat the evaluation steps described above. Here, we have to consider a much larger set of possible classes per entity (the 20 overall most frequent classes). The MFC baseline for these cases is the most frequent of the 20 classes for UNK tokens, while the random baseline chooses randomly from that set.

3.2 Generalization

Generalization measures how widely applicable the produced propositions are. A completely lexical approach, at one extreme, would turn each sentence into a separate proposition, thus achieving a generalization of 0%. At the other extreme, a model that produces only one proposition would generalize extremely well (but would fail to explain the data in any meaningful way). Both are of course not desirable.

[2]Unfortunately, if judged insensible, we can not infer whether our model used the wrong class despite better options, or whether we simply have not learned the correct label.

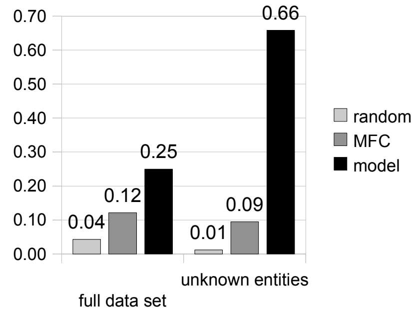

Figure 4: Generalization of models on the data sets

We define generalization as

$$g = 1 - \frac{|propositions|}{|sentences|} \tag{2}$$

The results in Figure 4 show that our model is capable of abstracting away from the lexical form, achieving a generalization rate of 25% for the full data set. The baseline approaches do significantly worse, since they are unable to detect similarities between lexically different examples, and thus create more propositions. Using a two-tailed t-test, the difference between our model and each baseline is statistically significant at $p < .001$.

Generalization on the unknown entity data set is even higher (65.84%). The difference between the model and the baselines is again statistically significant at $p < .001$. MFC always chooses the same class for UNK, regardless of context, and performs much worse. The random baseline chooses between 20 classes per entity instead of 3, and is thus even less general.

3.3 Normalized Entropy

Entropy is used in information theory to measure how predictable data is. 0 means the data is completely predictable. The higher the entropy of our propositions, the less well they explain the data. We are looking for models with low entropy. The extreme case of only one proposition has 0 entropy:

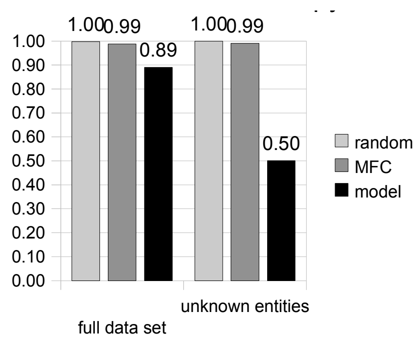

Figure 5: Entropy of models on the data sets

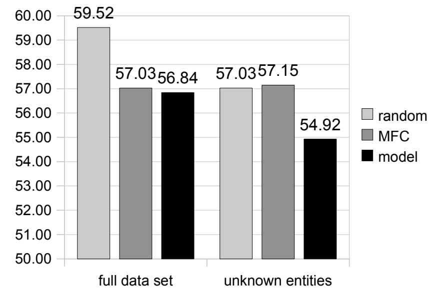

Figure 6: Perplexity of models on the data sets

we know exactly which sentences are produced by the proposition.

Entropy is directly influenced by the number of propositions used by a system.[3] In order to compare different models, we thus define normalized entropy as

$$H_N = \frac{-\sum_{i=0}^{n} P_i \cdot \log P_i}{\log n} \tag{3}$$

where P_i is the coverage of the proposition, or the percentage of sentences explained by it, and n is the number of distinct propositions.

The entropy of our model on the full data set is relatively high with 0.89, see Figure 5. The best entropy we can hope to achieve given the number of propositions and sentences is actually 0.80 (by concentrating the maximum probability mass in one proposition). The model thus does not perform as badly as the number might suggest. The entropy of our model on unseen data is better, with 0.50 (best possible: 0.41). This might be due to the fact that we considered more classes for UNK than for regular entities.

3.4 Perplexity

Since we assume that propositions are valid sentences in our domain, good propositions should have a higher probability than bad propositions in a language model. We can compute this using perplexity:[4]

$$perplexity(data) = 2^{\frac{-\log P(data)}{n}} \tag{4}$$

where $P(data)$ is the product of the proposition probabilities, and n is the number of propositions. We use the uni-, bi-, and trigram counts of the GoogleGrams corpus (Brants and Franz, 2006) with simple interpolation to compute the probability of each proposition.

The results in Figure 6 indicate that the propositions found by the model are preferable to the ones found by the baselines. As would be expected, the sensibility judgements for MFC and model[5] (Tables 1 and 2, Section 3.5) are perfectly anti-correlated (correlation coefficient -1) with the perplexity for these systems in each data set. However, due to the small sample size, this should be interpreted cautiously.

3.5 Sensibility and Label Accuracy

In unsupervised training, the model with the best data likelihood does not necessarily produce the best label accuracy. We evaluate label accuracy by presenting subjects with the propositions we obtained from the Viterbi decoding of the corpus, and ask them to rate their sensibility. We compare the different systems by computing sensibility as the percentage of propositions judged sensible for each system. Since the underlying probability distributions are quite different, we weight the sensibility judgement for each proposition by the likelihood of that proposition. We report results for both aggregate

[3]Note that how many classes we consider per entity influences how many propositions are produced (cf. Section 2.2), and thus indirectly puts a bound on entropy.

[4]Perplexity also quantifies the uncertainty of the resulting propositions, where 0 perplexity means no uncertainty.

[5]We did not collect sensibility judgements for the random baseline.

		100 most frequent		random		combined	
Data set	System	agg	maj	agg	maj	agg	maj
full	baseline	90.16	92.13	69.35	**70.57**	88.84	90.37
	model	**94.28**	**96.55**	**70.93**	70.45	**93.06**	**95.16**

Table 1: Percentage of propositions derived from labeling the full data set that were judged sensible

		100 most frequent		random		combined	
Data set	System	agg	maj	agg	maj	agg	maj
unknown	baseline	51.92	51.51	32.39	28.21	50.39	49.66
	model	**66.00**	**69.57**	**48.14**	**41.74**	**64.83**	**67.76**

Table 2: Percentage of propositions derived from labeling unknown entities that were judged sensible

sensibility (using the total number of individual answers), and majority sensibility, where each proposition is scored according to the majority of annotators' decisions.

The model and baseline propositions for the full data set are both judged highly sensible, achieving accuracies of 96.6% and 92.1% (cf. Table 1). While our model did slightly better, the differences are not statistically significant when using a two-tailed test. The propositions produced by the model from unknown entities are less sensible (67.8%), albeit still significantly above chance level, and the baseline propositions for the same data set ($p < 0.01$). Only 49.7% propositions of the baseline were judged sensible (cf. Table 2).

3.5.1 Annotation Task

Our model finds $250,169$ distinct propositions, the MFC baseline $293,028$. We thus have to restrict ourselves to a subset in order to judge their sensibility. For each system, we sample the 100 most frequent propositions and 100 random propositions found for both the full data set and the unknown entities[6] and have 10 annotators rate each proposition as sensible or insensible. To identify and omit bad annotators ('spammers'), we use the method described in Section 3.5.2, and measure inter-annotator agreement as described in Section 3.5.3. The details of this evaluation are given below, the results can be found in Tables 1 and 2.

The 200 propositions from each of the four systems (model and baseline on both full and unknown data set), contain 696 distinct propositions. We break these up into 70 batches (Amazon Turk annotation HIT pages) of ten propositions each. For each proposition, we request 10 annotators. Overall, 148 different annotators participated in our annotation. The annotators are asked to state whether each proposition represents a sensible statement about American Football or not. A proposition like "*Quarterbacks can throw passes to receivers*" should make sense, while "*Coaches can intercept teams*" does not. To ensure that annotators judge sensibility and not grammaticality, we format each proposition the same way, namely pluralizing the nouns and adding "can" before the verb. In addition, annotators can state whether a proposition sounds odd, seems ungrammatical, is a valid sentence, but against the rules (e.g., "*Coaches can hit players*") or whether they do not understand it.

3.5.2 Spammers

Some (albeit few) annotators on Mechanical Turk try to complete tasks as quickly as possible without paying attention to the actual requirements, introducing noise into the data. We have to identify these spammers before the evaluation. One way is to include tests. Annotators that fail these tests will be excluded. We use a repetition (first and last question are the same), and a truism (annotators answering "no" either do not know about football or just answered randomly). Alternatively, we can assume that good annotators, who are the majority, will exhibit similar behavior to one another, while spam-

[6]We omit the random baseline here due to size issues, and because it is not likely to produce any informative comparison.

mers exhibit a deviant answer pattern. To identify those outliers, we compare each annotator's agreement to the others and exclude those whose agreement falls more than one standard deviation below the average overall agreement.

We find that both methods produce similar results. The first method requires more careful planning, and the resulting set of annotators still has to be checked for outliers. The second method has the advantage that it requires no additional questions. It includes the risk, though, that one selects a set of bad annotators solely because they agree with one another.

3.5.3 Agreement

measure	100 most frequent	random	combined
agreement	0.88	0.76	0.82
κ	0.45	0.50	0.48
G-index	0.66	0.53	0.58

Table 3: Agreement measures for different samples

We use inter-annotator agreement to quantify the reliability of the judgments. Apart from the simple agreement measure, which records how often annotators choose the same value for an item, there are several statistics that qualify this measure by adjusting for other factors. One frequently used measure, Cohen's κ, has the disadvantage that if there is prevalence of one answer, κ will be low (or even negative), despite high agreement (Feinstein and Cicchetti, 1990). This phenomenon, known as the κ paradox, is a result of the formula's adjustment for chance agreement. As shown by Gwet (2008), the true level of actual chance agreement is realistically not as high as computed, resulting in the counterintuitive results. We include it for comparative reasons. Another statistic, the G-index (Holley and Guilford, 1964), avoids the paradox. It assumes that expected agreement is a function of the number of choices rather than chance. It uses the same general formula as κ,

$$\frac{(P_a - P_e)}{(1 - P_e)} \tag{5}$$

where P_a is the actual raw agreement measured, and P_e is the expected agreement. The difference with κ is that P_e for the G-index is defined as $P_e = 1/q$, where q is the number of available categories, instead of expected chance agreement. Under most conditions, G and κ are equivalent, but in the case of high raw agreement and few categories, G gives a more accurate estimation of the agreement. We thus report raw agreement, κ, and G-index.

Despite early spammer detection, there are still outliers in the final data, which have to be accounted for when calculating agreement. We take the same approach as in the statistical spammer detection and delete outliers that are more than one standard deviation below the rest of the annotators' average.

The raw agreement for both samples combined is 0.82, $G = 0.58$, and $\kappa = 0.48$. The numbers show that there is reasonably high agreement on the label accuracy.

4 Related Research

The approach we describe is similar in nature to unsupervised verb argument selection/selectional preferences and semantic role labeling, yet goes beyond it in several ways. For semantic role labeling (Gildea and Jurafsky, 2002; Fleischman et al., 2003), classes have been derived from FrameNet (Baker et al., 1998). For verb argument detection, classes are either semi-manually derived from a repository like WordNet, or from NE taggers (Pardo et al., 2006; Fan et al., 2010). This allows for domain-independent systems, but limits the approach to a fixed set of oftentimes rather inappropriate classes. In contrast, we derive the level of granularity directly from the data.

Pre-tagging the data with NE classes before training comes at a cost. It lumps entities together which can have very different classes (i.e., all people become labeled as PERSON), effectively allowing only one class per entity. Etzioni et al. (2005) resolve the problem with a web-based approach that learns hierarchies of the NE classes in an unsupervised manner. We do not enforce a taxonomy, but include statistical knowledge about the distribution of possible classes over each entity by incorporating a prior distribution $P(class, entity)$. This enables us to generalize from the lexical form without restricting ourselves to one class per entity, which helps to better fit the data. In addition, we can distinguish several classes for each entity, depending on the context (e.g., *winner* vs. *quarterback*). Ritter et al. (2010) also use an unsupervised model to derive selectional predicates from unlabeled text. They do not assign classes altogether, but group similar predicates and arguments into unlabeled clusters using LDA. Brody (2007) uses a very similar methodology to establish relations between clauses and sentences, by clustering simplified propositions.

Peñas and Hovy (2010) employ syntactic patterns to derive classes from unlabeled data in the context of LbR. They consider a wider range of syntactic structures, but do not include a probabilistic model to label new data.

5 Conclusion

We use an unsupervised model to infer domain-specific classes from a corpus of 1.4m unlabeled sentences, and applied them to learn 250k propositions about American football. Unlike previous approaches, we use automatically extracted classes with a probability distribution over entities to allow for context-sensitive selection of appropriate classes.

We evaluate both the model qualities and sensibility of the resulting propositions. Several measures show that the model has good explanatory power and generalizes well, significantly outperforming two baseline approaches, especially where the possible classes of an entity can only be inferred from the context.

Human subjects on Amazon's Mechanical Turk judged up to 96.6% of the propositions for the full data set, and 67.8% for data containing unseen entities as sensible. Inter-annotator agreement was reasonably high ($agreement = 0.82$, $G = 0.58$, $\kappa = 0.48$).

The probabilistic model and the extracted propositions can be used to enrich texts and support post-parsing inference for question answering. We are currently applying our method to other domains.

Acknowledgements

We would like to thank David Chiang, Victoria Fossum, Daniel Marcu, and Stephen Tratz, as well as the anonymous ACL reviewers for comments and suggestions to improve the paper. Research supported in part by Air Force Contract FA8750-09-C-0172 under the DARPA Machine Reading Program.

References

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet Project. In *Proceedings of the 17th international conference on Computational linguistics-Volume 1*, pages 86–90. Association for Computational Linguistics Morristown, NJ, USA.

Thorsten Brants and Alex Franz, editors. 2006. *The Google Web 1T 5-gram Corpus Version 1.1*. Number LDC2006T13. Linguistic Data Consortium, Philadelphia.

Samuel Brody. 2007. Clustering Clauses for High-Level Relation Detection: An Information-theoretic Approach. In *Annual Meeting-Association for Computational Linguistics*, volume 45, page 448.

Marie-Catherine De Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*. Citeseer.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Oren Etzioni, Michael Cafarella, Doug. Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. 2005. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134.

James Fan, David Ferrucci, David Gondek, and Aditya Kalyanpur. 2010. Prismatic: Inducing knowledge from a large scale lexicalized relation resource. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 122–127, Los Angeles, California, June. Association for Computational Linguistics.

Alvan R. Feinstein and Domenic V. Cicchetti. 1990. High agreement but low kappa: I. the problems of two paradoxes. *Journal of Clinical Epidemiology*, 43(6):543–549.

Michael Fleischman, Namhee Kwon, and Eduard Hovy. 2003. Maximum entropy models for FrameNet classification. In *Proceedings of EMNLP*, volume 3.

Danies Gildea and Dan Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.

Kilem Li Gwet. 2008. Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics-Volume 2*, pages 539–545. Association for Computational Linguistics.

Jasper Wilson Holley and Joy Paul Guilford. 1964. A Note on the G-Index of Agreement. *Educational and Psychological Measurement*, 24(4):749.

Rutu Mulkar-Mehta, James Allen, Jerry Hobbs, Eduard Hovy, Bernardo Magnini, and Christopher Manning, editors. 2010. *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*. Association for Computational Linguistics, Los Angeles, California, June.

Thiago Pardo, Daniel Marcu, and Maria Nunes. 2006. Unsupervised Learning of Verb Argument Structures. *Computational Linguistics and Intelligent Text Processing*, pages 59–70.

Anselmo Peñas and Eduard Hovy. 2010. Semantic enrichment of text with background knowledge. In *Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading*, pages 15–23, Los Angeles, California, June. Association for Computational Linguistics.

Simone Paolo Ponzetto and Roberto Navigli. 2010. Knowledge-rich Word Sense Disambiguation rivaling supervised systems. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1522–1531. Association for Computational Linguistics.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Uppsala, Sweden, July. Association for Computational Linguistics.

Evan Sandhaus, editor. 2008. *The New York Times Annotated Corpus*. Number LDC2008T19. Linguistic Data Consortium, Philadelphia.

Rion Snow, Brendan O'Connor, Dan Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast—but is it good? Evaluating non-expert annotations for natural language tasks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 254–263. Association for Computational Linguistics.

Stephanie Strassel, Dan Adams, Henry Goldberg, Jonathan Herr, Ron Keesing, Daniel Oblinger, Heather Simpson, Robert Schrag, and Jonathan Wright. 2010. The DARPA Machine Reading Program-Encouraging Linguistic and Reasoning Research with a Series of Reading Tasks. In *Proceedings of LREC 2010*.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM.