Entity Extraction via Ensemble Semantics

Marco Pennacchiotti
Yahoo! Labs
Sunnyvale, CA, 94089
pennac@yahoo-inc.com

Patrick Pantel
Yahoo! Labs
Sunnyvale, CA, 94089
ppantel@yahoo-inc.com

Abstract

Combining information extraction systems yields significantly higher quality resources than each system in isolation. In this paper, we generalize such a mixing of sources and features in a framework called *Ensemble Semantics*. We show very large gains in entity extraction by combining state-of-the-art distributional and pattern-based systems with a large set of features from a webcrawl, query logs, and Wikipedia. Experimental results on a web-scale extraction of actors, athletes and musicians show significantly higher mean average precision scores (29% gain) compared with the current state of the art.

1 Introduction

Mounting evidence shows that combining information sources and information extraction algorithms leads to improvements in several tasks such as fact extraction (Paşca et al., 2006), open-domain IE (Talukdar et al., 2008), and entailment rule acquisition (Mirkin et al., 2006). In this paper, we show large gains in entity extraction by combining state-of-the-art distributional and pattern-based systems with a large set of features from a 600 million document webcrawl, one year of query logs, and a snapshot of Wikipedia. Further, we generalize such a mixing of sources and features in a framework called *Ensemble Semantics*.

Distributional and pattern-based extraction algorithms capture aspects of paradigmatic and syntagmatic dimensions of semantics, respectively, and are believed to be quite complementary. Paşca et al. (2006) showed that filtering facts, extracted by a pattern-based system, according to their arguments' distributional similarity with seed facts yielded large precision gains. Mirkin et al. (2006) showed similar gains on the task of acquiring lexical entailment rules by exploring a supervised combination of distributional and pattern-based algorithms using an ML-based SVM classifier.

This paper builds on the above work, by studying the impact of various sources of features external to distributional and pattern-based algorithms, on the task of entity extraction. Mirkin et al.'s results are corroborated on this task and large and significant gains over this baseline are obtained by incorporating 402 features from a webcrawl, query logs and Wikipedia. We extracted candidate entities for the classes *Actors*, *Athletes* and *Musicians* from a webcrawl using a variant of Paşca et al.'s (2006) pattern-based engine and Pantel et al.'s (2009) distributional extraction system. A gradient boosted decision tree is used to learn a regression function over the feature space for ranking the candidate entities. Experimental results show 29% gains (19% nominal) in mean average precision over Mirkin et al.'s method and 34% gains (22% nominal) in mean average precision over an unsupervised baseline similar to Paşca et al.'s method. Below we summarize the contributions of this paper:

- We explore the hypothesis that although distributional and pattern-based algorithms are complementary, they do not exhaust the semantic space; other sources of evidence can be leveraged to better combine them;
- We model the mixing of knowledge sources and features in a novel and general information extraction framework called *Ensemble Semantics*; and
- Experiments over an entity extraction task show that our model achieves large and significant gains over state-of-the-art extractors. A detailed analysis of feature correlations and interactions shows that query log and webcrawl features yield the highest gains, but easily accessible Wikipedia features also improve over current state-of-the-art systems.

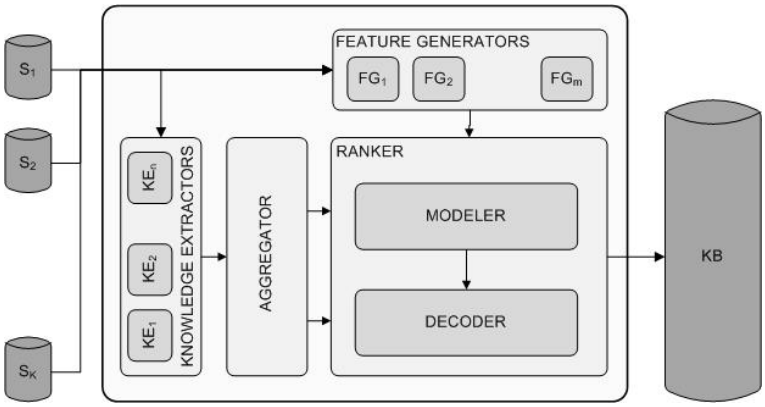

Figure 1: The *Ensemble Semantics* framework for information extraction.

The remainder of this paper is organized as follows. In the next section, we present our Ensemble Semantics framework and outline how various information extraction systems can be cast into the framework. Section 3 then presents our entity extraction system as an instance of Ensemble Semantics, comparing and contrasting it with previous information extraction systems. Our experimental methodology and analysis is described in Section 4 and shows empirical evidence that our extractor significantly outperforms prior art. Finally, Section 5 concludes with a discussion and future work.

2 Ensemble Semantics

Ensemble Semantics (ES) is a general framework for modeling information extraction algorithms that combine multiple sources of information and multiple extractors. The ES framework allows to:

- Represent multiple sources of knowledge and multiple extractors of that knowledge;
- Represent multiple sources of features;
- Integrate both rule-based and ML-based knowledge ranking algorithms; and
- Model previous information extraction systems (i.e., backwards compatibility).

2.1 ES Framework

ES can be instantiated to extract various types of knowledge such as entities, facts, and lexical entailment rules. It can also be used to better understand the commonalities and differences between existing information extraction systems.

After presenting the framework in the next section, Section 2.2 shows how previous information extraction algorithms can be cast into ES. In Section 3 we describe our novel entity extraction algorithm based on ES.

The ES framework is illustrated in Figure 1. It decomposes the process of information extraction into the following components:

Sources (S): textual repositories of information, either structured (e.g., a database such as DBpedia), semi-structured (e.g., Wikipedia Infoboxes or HTML tables) or unstructured (e.g., news articles or a webcrawl).

Knowledge Extractors (KE): algorithms responsible for extracting candidate instances such as entities or facts. Examples include fact extraction systems such as (Cafarella et al., 2005) and entity extraction systems such as (Paşca, 2007).

Feature Generators (FG): methods that extract evidence (features) of knowledge in order to decide which candidate instances extracted from KEs are correct. Examples include capitalization features for named entity extractors, and the distributional similarity matrix used in (Paşca et al., 2006) for filtering facts.

Aggregator (A). A module collecting and assembling the instances coming from the different extractors. This module keeps the *footprint* of each instance, i.e. the number and the type of the KEs that extracted the instance. This information can be used by the Ranker module to build a ranking strategy, as described below.

Ranker (R): a module for ranking the knowledge instances returned from KEs using the features generated by FGs. Ranking algorithms may be rule-based (e.g., the one using a threshold on distributional similarity in (Paşca et al., 2006)) or ML-based (e.g., the SVM model in (Mirkin et al., 2006) for combining pattern-based and distributional features).

The Ranker is composed of two sub-modules: the Modeler and the Decoder. The *Modeler* is responsible for creating the model which ranks the candidate instances. The *Decoder* collects the candidate instances from the Aggregator, and applies the model to produce the final ranking.

In rule-based systems, the Modeler corresponds to a set of manually crafted or automatically induced rules operating on the features (e.g. a combination of thresholds). In ML-based systems, it is an actual machine learning algorithm, that takes as input a set of labeled training instances, and builds the model according to their features. Training instances can be obtained as a subset of those collected by the Aggregator, or from some external resource. In many cases, training instances are manually labeled by human experts, through a long and costly editorial process.

Information sources (S) serve as inputs to the system. Some sources will serve as sources for knowledge extractors to generate candidate instances, some will serve as sources for feature generators to generate features or evidence of knowledge, and some will serve as both.

2.2 Related Work

To date, most information extraction systems rely on a model composed of a single source S, a single extractor KE and a single feature generator FG. For example, many classic relation extraction systems (Hearst, 1992; Riloff and Jones, 1999; Pantel and Pennacchiotti, 2006; Paşca et al., 2006) are based on a single pattern-based extractor KE, which is seeded with a set of patterns or instances for a given relation (e.g. the pattern *'X starred in Y'* for the *act-in* relation). The extractor then iteratively extracts new instances until a stop condition is met. The resulting extractor scores are proposed by FG as a feature. The *Ranker* simply consists of a sorting function on the feature from FG.

Systems such as the above that do not consist of multiple sources, knowledge extractors or feature generators are not considered Ensemble Semantics models, even though they can be cast into the framework. Recently, some researchers have explored more complex systems, having multiple sources, extractors and feature generators. Below we show examples and describe how they map as Ensemble Semantics systems. We use this characterization to clearly outline how our proposed entity extraction system, proposed in Section 3, differs from previous work.

Talukdar et al. (2008) present a weakly-supervised system for extracting large sets of class-instance pairs using two knowledge extractors: a pattern-based extractor supported by distributional evidence, which harvests candidate pairs from a Web corpus; and a table extractor that harvests candidates from Web tables. The *Ranker* uses graph random walks to combine the information of the two extractors and output the final list. The authors show large improvements in coverage with little precision loss.

Mirkin et al. (2006) introduce a machine learning system for extracting lists of lexical entailments (e.g. 'government' → 'organization'). They rely on two knowledge extractors, operating on a same large textual source: a pattern-based extractor, leveraging the Hearst (1992) *is-a* patterns; and a distributional extractor applied to a set of entailment seeds. Candidate instances are passed to an SVM *Ranker*, which uses features stemming from the two extractors, to decide which instances are output in the final list. The authors report a +9% increase in F-measure over a rule-based system that takes the union of the instances extracted by the two modules.

Other examples include the system for taxonomic-relation extraction by Cimiano et al. (2005), using a pool of feature generators based on pattern-based, distributional and WordNet techniques; and Paşca and Van Durme's (2008) system that uses a Web corpus and query logs to extract semantic classes and their attributes.

Similarly to these methods, our proposed entity extractor (Section 3) utilizes multiple sources and extractors. A key difference of our method lies in the Feature Generator module. We propose several generators resulting in 402 features extracted from Web pages, query logs and Wikipedia articles. The use of these features results in dramatic performance improvements, reported in Section 4.

3 ES for Entity Extraction

Entity extraction is a fundamental task in NLP responsible for extracting instances of semantic classes (e.g., *'Brad Pitt'* and *'Tom Hanks'* are instances of the class *Actors*). It forms a building block for various NLP tasks such as ontology learning (Cimiano and Staab, 2004) and co-reference resolution (Mc Carthy and Lehn-

Family	*Type*		*Features*
Web (w)	Frequency	(wF)	term frequency; document frequency; term frequency as noun phrase
	Pattern	(wP)	confidence score returned by KE_{pat}; pmi with the 100 most reliable patterns used by KE_{pat}
	Distributional	(wD)	distributional similarity with the centroid in KE_{dis}; distributional similarities with each seed in $\mathcal{S}$
	Termness	(wT)	ratio between term frequency as noun phrase and term frequency; pmi between internal tokens of the instance; capitalization ratio
Query log (q)	Frequency	(qF)	number of queries matching the instance; number of queries containing the instance
	Co-occurrence	(qC)	query log pmi with any seed in $\mathcal{S}$
	Pattern	(qP)	pmi with a set of trigger words $\mathcal{T}$ (i.e., the 10 words in the query logs with highest pmi with $\mathcal{S}$)
	Distributional	(qD)	distributional similarity with $\mathcal{S}$ (vector coordinates consist of the instance's pmi with the words in $\mathcal{T}$)
	Termness	(qT)	ratio between the two frequency features F
Web table (t)	Frequency	(tF)	table frequency
	Co-occurrence	(tC)	table pmi with $\mathcal{S}$; table pmi with any seed in $\mathcal{S}$
Wikipedia (k)	Frequency	(kF)	term frequency
	Co-occurrence	(kC)	pmi with any seed in $\mathcal{S}$
	Distributional	(kD)	distributional similarity with $\mathcal{S}$

Table 1: Feature space describing each candidate instance ($\mathcal{S}$ indicates the set of seeds for a given class).

ert, 2005). Search engines such as Yahoo, Live, and Google collect large sets of entities (Paşca, 2007; Chaudhuri et al., 2009) to better interpret queries (Tan and Peng, 2006), to improve query suggestions (Cao et al., 2008) and to understand query intents (Hu et al., 2009). Entity extraction differs from the similar task of named entity extraction, in that classes are more fine-grained and possibly overlapping.

Below, we propose a new method for entity extraction built on the ES framework (Section 3.1). Then, we comment on related work in entity extraction (Section 3.2).

3.1 ES Entity Extraction Model

In this section, we propose a novel entity extraction model following the Ensemble Semantics framework presented in Section 2. The sources of our systems can come from any textual corpus. In our experiments (described in Section 4.1), we extracted entities from a large crawl of the Web, and generated features from this crawl as well as query logs and Wikipedia.

3.1.1 Knowledge extractors

Our system relies on two knowledge extractors: one pattern-based and the other distributional.

Pattern-based extractor (KE_{pat}). We reimplemented Paşca et al.'s (2006) state-of-the-art web-scale fact extractor, which, given seed instances of a binary relation, finds instances of that relation. We extract entities of a class, such as *Actors*, by instantiating typical relations involving that class such as *act-in(Actor, Movie)*. We instantiate such relations instead of the classical *is-a* patterns since these have been shown to bring in too many false positives, see (Pantel and Pennacchiotti, 2006) for a discussion of such generic patterns. The extractor's confidence score for each instance is used by the *Ranker* to score the entities being extracted. Section 4.1 lists the system parameters we used in our experiments.

Distributional extractor (KE_{dis}). We use Pantel et al.'s (2009) distributional entity extractor. For each noun in our source corpus, we build a context vector consisting of the noun chunks preceding and following the target noun, scored using pointwise mutual information (pmi). Given a small set of seed entities $\mathcal{S}$ of a class, the extractor computes the centroid of the seeds' context vectors as a geometric mean, and then returns all nouns whose similarity with the centroid exceeds a threshold τ (using the cosine measure between the context vectors). Full algorithmic details are presented in (Pantel et al., 2009). Section 4.1 lists the threshold and text preprocessing algorithms used in our experiments.

The *Aggregator* simply takes a union of the entities discovered by the two extractors.

3.1.2 Feature generators

Our model includes four feature generators, which compute a total of 402 features (full set described in Table 1). Each generator extracts from a specific source a *feature family*, as follows:

- *Web (w)*: a body of 600 million documents

crawled from the Web at Yahoo! in 2008;

- *Query logs (q)*: one year of web search queries issued to the Yahoo! search engine;
- *Web tables*: all HTML inner tables extracted from the above *Web* source; and
- *Wikipedia*: an official Wikipedia dump from February 2008, consisting of about 2 million articles.

Feature families are further subclassified into five types: *frequency (F)* (frequency-based features); *co-occurrence (C)* (features capturing first order co-occurrences between an instance and class seeds); *distributional (D)* (features based on the distributional similarity between an instance and class seeds); *pattern (P)* (features indicating class-specific lexical pattern matches); and *termness (T)* (features used to distinguish well-formed terms such as *'Brad Pitt'* from ill-formed ones such as *'with Brad Pitt'*). The seeds $\mathcal{S}$ used in many of the feature families are the same seeds used by the KE_{pat} extractor, described in Section 3.1.1.

The different seed families are designed to capture different semantic aspects: paradigmatic (D), syntagmatic (C and P), popularity (F), and term cohesiveness (T).

3.1.3 ML-based Ranker

Our *Modeler* adopts a supervised ML regression model. Specifically, we use a Gradient Boosted Decision Tree regression model - GBDT (Friedman, 2001), which consists of an ensemble of decision trees, fitted in a forward step-wise manner to current residuals. Friedman (2001) shows that by drastically easing the problem of overfitting on training data (which is common in boosting algorithms), GDBT competes with state-of-the-art machine learning algorithms such as SVM (Friedman, 2006) with much smaller resulting models and faster decoding time. The model is trained on a manually annotated random sample of entities taken from the *Aggregator*, using the features generated by the four generators presented in Section 3.1.2. The *Decoder* then ranks each entity according to the trained model.

3.2 Related Work

Entity extraction systems follow two main approaches: pattern-based and distributional. The *pattern-based approach* leverages lexico-syntactic patterns to extract instances of a given class. Most commonly used are *is-a* pattern families such as those first proposed by Hearst (1992) (e.g., *'Y such as X'* for matching *'actors such as Brad Pitt'*). Minimal supervision is used in the form of small sets of manually provided seed patterns or seed instances. This approach is very common in both the NLP and Semantic Web communities (Cimiano and Staab, 2004; Cafarella et al., 2005; Pantel and Pennacchiotti, 2006; Paşca et al., 2006).

The *distributional approach* uses contextual evidence to model the instances of a given class, following the distributional hypothesis (Harris, 1964). Weakly supervised, these methods take a small set of seed instances (or the class label) and extract new instances from noun phrases that are most similar to the seeds (i.e., that share similar contexts). Following Lin (1998), example systems include Fleischman and Hovy (2002), Cimiano and Volker (2005), Tanev and Magnini (2006), and Pantel et al. (2009).

4 Experimental Evaluation

This section reports our experiments, showing the effectiveness of our entity extraction system and the importance of our different feature families.

4.1 Experimental Setup

Evaluated classes. We evaluate our system over three classes: *Actors* (movie, tv and stage actors); *Athletes* (professional and amateur); *Musicians* (singers, musicians, composers, bands, and orchestras)

System setup. We instantiated our knowledge extractors, KE_{pat} and KE_{dis} from Section 3.1.1, over our *Web* crawl of 600 million documents (see Section 3.1.2). The documents were preprocessed using Brill's POS-tagger (Brill, 1995) and the Abney's chunker (Abney, 1991). For $\boldsymbol{KE_{dis}}$, context vectors are extracted for noun phrases recognized as NP-chunks with removed modifiers. The vector space includes the 250M most frequent noun chunks in the corpus. KE_{dis} returns as instances all noun phrases having a similarity with the seeds' centroid above $\tau = 0.005$[1]. The sets of seeds $\mathcal{S}$ for KE_{dis} include 10, 24 and 10 manually chosen instances for respectively the *Actors*, *Athletes* and *Musicians* classes[2]. The sets of seeds $\mathcal{P}$ for KE_{pat}

[1] Experimentally set on an independent development set.

[2] The higher number of seeds for *Athletes* is chosen to cover different sports.

Dataset	*Actors*	*Athletes*	*Musicians*
KE_{pat}	58,005	40,816	125,657
KE_{dis}	72,659	24,380	24,593
$KE_{pat} \cup KE_{dis}$	113,245	61,709	142,694
$KE_{pat} \cap KE_{dis}$	17,419	3,487	7,556
$\mathcal{R}$	500	500	500
	P=80	*P=258*	*P=134*
	N=420	*N=242*	*N=366*

Table 2: Number of extracted instances and the sample sizes (P and N indicate positive and negative annotations).

include 11, 8 and 9 pairs respectively for the *Actors* (relation *acts-in*), *Athletes* (relation *plays-for*) and *Musicians* (relation *part-of-band*) classes. Table 6 lists all seeds for both KE_{dis} and KE_{pat}. The ***GBDT ranker*** uses an ensemble of 300 trees.[3]

Goldset Preparation. The number of instances extracted by KE_{pat} and KE_{dis} for each class is reported in Table 2. For each class, we extract a random sample $\mathcal{R}$ of 500 instances from $KE_{pat} \cup KE_{dis}$. A pool of 10 paid expert editors annotated the instances of each class in $\mathcal{R}$ as positive or negative. Inter-annotator overlap was 0.88. Uncertain instances were manually adjudicated by a separate paid expert editor, yielding a gold standard dataset for each class.

Evaluation Metrics. Entity extraction performance is evaluated using the *average precision* (AP) statistic, a standard information retrieval measure for evaluating ranking algorithms, defined as:

$$AP(L) = \frac{\sum_{i=1}^{|L|} P(i) \cdot corr(i)}{\sum_{i=1}^{|L|} corr(i)} \qquad (1)$$

where L is a ranked list produced by an extractor, $P(i)$ is the precision of L at rank i, and *corr(i)* is 1 if the instance at rank i is correct, and 0 otherwise. AP is computed over $\mathcal{R}$ for each class.

We also evaluate the *coverage*, i.e. the percentage of instances extracted by a system wrt those extracted by all systems.

In order to accurately compute statistical significance, our experiments are performed using 10-fold cross validation.

Baselines and comparisons. We compare our proposed ES entity extractor, using different feature configurations, with state-of-the-art systems (referred to as baselines *B** below):

[3]GBDT model parameters were experimentally set on an independent development set as follows: trees=300, shrinkage=0.01, max_nodes_per_tree=12, sample_rate=0.5.

System	Actors		Athletes		Musicians	
	AP	***Cov***	***AP***	***Cov***	***AP***	***Cov***
B1	0.729	51.2%	0.616	66.1%	0.570	88.1%
B2	0.618	64.1%	0.687	39.5%	0.681	17.2%
B3	0.676	100%	0.664	100%	0.576	100%
B4	0.715	100%	0.697	100%	0.579	100%
ES-all	**0.860**‡	100%	**0.915**‡	100%	**0.788**‡	100%

Table 3: Average precision (AP) and coverage (Cov) results for our proposed system *ES-all* and the baselines. ‡ indicates AP statistical significance at the 0.95 level wrt all baselines.

ES-all. Our ES system, using KE_{pat} and KE_{dis}, the full set of feature families described in Section 3.1.2, and the GBDT ranker.

B1. KE_{pat} alone, a state-of-the-art pattern-based extractor reimplementing (Paşca et al., 2006), where the *Ranker* assigns scores to instances using the confidence score returned by KE_{pat}.

B2. KE_{dis} alone, a state-of-the-art distributional system implementing (Pantel et al., 2009), where the *Ranker* assigns scores to instances using the similarity score returned by KE_{dis} alone.

B3. A rule-based ES system, combining *B1* and *B2*. This system uses both KE_{pat} and KE_{dis} as extractors, and a *Ranker* that assigns scores to instances according to the sum of their normalized confidence scores.

B4. A state-of-the-art machine learning system based on (Mirkin et al., 2006). This ES system uses KE_{pat} and KE_{dis} as extractors. The *Ranker* is a GBDT regression model, using the full sets of features derived from the two extractors, i.e., wP and wD (see Table 1). GBDT parameters are set as for our proposed *ES-all* system.

4.2 Experimental Results

Table 3 summarizes the average-precision (AP) and coverage results for our *ES-all* system and the baselines. Figure 2 reports the precision at each rank for the *Athletes* class (the other two classes follow similar trends). Table 6 lists the top-10 entities discovered for each class on one test fold. *ES-all* outperforms all baselines in AP (all results are statistically significant at the 0.95 level), offering at the same time full coverage[4].

[4]Recall that coverage is reported relative to all instances retrieved by extractors KE_{pat} and KE_{dis}.

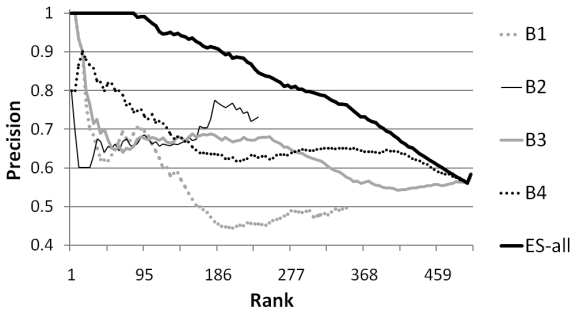

Figure 2: Precision at rank for the different systems on the *Athletes* class.

System	*AP*			*MAP*
	Actors	**Athletes**	**Musicians**	
B3	0.676	0.664	0.576	0.639
B4	0.715	0.697	0.579	0.664
B4+w	$0.813^{\ddagger}$	$0.908^{\ddagger}$	$0.724^{\ddagger}$	$0.815^{\ddagger}$
B4+q	$0.815^{\ddagger}$	$0.905^{\ddagger}$	$0.743^{\ddagger}$	$0.821^{\ddagger}$
B4+t	$0.784^{\dagger}$	$0.825^{\ddagger}$	$0.727^{\ddagger}$	$0.779^{\ddagger}$
B4+k	$0.776^{\dagger}$	$0.825^{\ddagger}$	0.624	$0.741^{\dagger}$
B4+w+q	$0.835^{\ddagger}$	$0.915^{\ddagger}$	$0.758^{\ddagger}$	$0.836^{\ddagger}$
B4+w+t	$0.840^{\ddagger}$	$0.906^{\ddagger}$	$0.774^{\ddagger}$	$0.840^{\ddagger}$
B4+w+k	$0.814^{\ddagger}$	$0.903^{\ddagger}$	$0.725^{\ddagger}$	$0.814^{\ddagger}$
B4+q+t	$0.847^{\ddagger}$	$0.910^{\ddagger}$	$0.774^{\ddagger}$	$0.844^{\ddagger}$
B4+q+k	$0.832^{\ddagger}$	$0.906^{\ddagger}$	$0.748^{\ddagger}$	$0.829^{\ddagger}$
B4+t+k	$0.817^{\ddagger}$	$0.861^{\ddagger}$	$0.743^{\ddagger}$	$0.807^{\ddagger}$
B4+w+q+t	$0.846^{\ddagger}$	$\mathbf{0.917}^{\ddagger}$	$0.782^{\ddagger}$	$0.849^{\ddagger}$
B4+w+q+k	$0.841^{\ddagger}$	$0.916^{\ddagger}$	$0.756^{\ddagger}$	$0.838^{\ddagger}$
B4+w+t+k	$0.835^{\ddagger}$	$0.906^{\ddagger}$	$0.783^{\ddagger}$	$0.841^{\ddagger}$
Es-all	$\mathbf{0.860}^{\ddagger}$	$0.915^{\ddagger}$	$\mathbf{0.788}^{\ddagger}$	$\mathbf{0.854}^{\ddagger}$

Table 4: Overall AP results of the different feature configurations, compared to two baselines. † indicates statistical significance at the 0.95 level wrt *B3*. ‡ indicates statistical significance at 0.95 level wrt both *B3* and *B4*.

Our simple rule-based combination baseline, *B3*, leads to a large increase in coverage wrt the individual extractors alone (*B1* and *B2*) without significant impact on precision. The supervised ML-based combination baseline (*B4*) consistently improves AP across classes wrt the rule-based combination (*B3*), but without statistical significance. These results corroborate those found in (Mirkin et al., 2006), where this ML-based combination was reported to be significantly better than a rule-based one on the task of lexical entailment acquisition.

The large set of features adopted in *ES-all* accounts for a dramatic improvement in AP, indicating that existing state-of-the-art systems for entity extraction (reflected by our baselines strategies) are not making use of enough semantic cues. The adoption of evidence other than distributional and pattern-based, such as features coming from web documents, HTML tables and query logs, is here demonstrated to be highly valuable.

The above empirical claim can be grounded and corroborated by a deeper semantic analysis. From a semantic perspective, the above results translate in the observation that distributional and pattern-based evidence do not completely capture all semantic aspects of entities. Other evidence, such as popularity, term cohesiveness and co-occurrences capture other aspects. For instance, in one of our *Actors* folds, the *B3* system ranks the incorrect instance *'Tom Sellek'* (a misspelling of *'Tom Selleck'*) in 9^{th} position (out of 142), while *ES-all* lowers it to the 33^{rd} position, by relying on table-based features (intuitively, tables contain much fewer misspelling than running text). Other than misspellings, *ES-all* fixes errors that are either typical of distributional approaches, such as the inclusion of instances of other classes (e.g. the movie *'Someone Like You'* often appears in contexts similar to those of *actors*); errors typical of pattern-based approaches, such as incorrect instances highly-associated with an ambiguous pattern (e.g., the pattern *'X of the rock band Y'* for finding *Musicians* matched an incorrect instance *'song submission'*); or errors typical of both, such as the inclusion of common nouns (e.g. *'country music hall'*) or too generic last names (e.g. *'Johnson'*). *ES-all* successfully recovers all these error by using termness, co-occurrence and frequency features.

We also compare *ES-all* with a state-of-the-art random walk system (*RW*) presented by Talukdar et al. (2008) (see Section 2.2 for a description). As we could not reimplement the system, we report the following indirect comparison. *RW* was evaluated on five entity classes, one of which, *NFL players*, overlaps with our *Athletes* class. On this class, they report 0.95 precision on the top-100 ranked entities. Unfortunately, they do not report coverage or recall statistics, making the interpretation of this analysis difficult. In an attempt to compare *RW* with *ES-all*, we evaluated the precision of our top-100 *Athletes*, obtaining 0.99. Using a random sample of our extracted *Athletes*, we approximate the precision of the top-22,000 *Athletes* to be 0.97 ± 0.01 (at the 0.95 level).

4.3 Feature Analysis

Feature family analysis: Table 4 reports the average precision (AP) for our system using different feature family combinations (see Table 1). Column 1 reports the family combinations; columns

2-4 report AP for each class; and column 5 reports the mean-average-precision (*MAP*) across classes. In all configurations, except the k family alone, and along all classes, our system significantly outperforms (at the 0.95 level) the baselines.

Rows 3-6 report the performance of each feature family alone. w and t are consistently better than q and k, across all classes. k is shown to be the least useful family. This is mostly due to data sparseness, e.g., in our experiments almost 40% of the test instances in the *Actors* sample do not have any occurrence in Wikipedia. However, without access to richer resources such as a webcrawl or query logs, the features from k do indeed provide large gains over current baselines (on average +10.2% and +7.7% over *B3* and *B4*).

Rows 7-12 report results for combinations of two feature families. All combinations (except those with k) appear valuable, substantially increasing the single-family results in rows 3-6, indicating that combining different feature families (as suggested by the ES paradigm) is helpful. Second, it indicates that q, w and t convey complementary information, thus boosting the regression model when combined together. It is interesting to notice that *q+t* tends to be the best combination, surprising given that t alone did not show high performance (row 5). One would expect the combination *q+w* to outperform *q+t*, but the good performance of *q+t* is mainly due to the fact that these two families are more complementary than *q+w*. To verify this intuition, we compute the Spearman correlation coefficient r among the rankings produced by the different combinations. As expected, *q* and *w* have a higher correlation ($r = 0.82$) than *q* and *t* ($r = 0.67$) and *w* and *t* ($r = 0.66$), suggesting that q and w tend to subsume each other (i.e. no added information for the regression model).

Rows 13-15 report results for combinations of three feature families. As expected, the best combination is *q+w+t* with an average precision nearly identical to the full *ES-all* system. If one has access to Web or query log sources, then the value of the Wikipedia features tends to be subsumed by our web and query log features.

Feature by feature analysis: The feature families analyzed in the previous section consist of 402 features. For each trained GBDT model, one can inspect the resulting most important features (Friedman, 2001). Consistently, the two most important features for *ES-all* are, as expected, the confidence scores of KE_{pat} and KE_{dis}. This suggests that syntagmatic and paradigmatic information are most important in defining the semantics of entities. Also very important, in third position, is a feature from qT, namely the ratio between the number of queries matching the instance and the number of queries containing it as a substring. This feature is a strong indicator of termness.

System	*AP*			*MAP*
	Actors	**Athletes**	**Musicians**	
B4	0.715	0.697	0.579	0.664
B4+**w**	**0.813**	**0.908**	**0.724**	**0.815**
B4+wF	0.798	0.865	0.679	0.781
B4+wT	0.806	0.891	0.717	0.805
B4+**t**	**0.784**	**0.825**	**0.727**	**0.779**
B4+tF	0.760	0.802	0.701	0.781
B4+tC	0.771	0.815	0.718	0.805
B4+**q**	**0.815**	**0.905**	**0.743**	**0.821**
B4+qF	0.786	0.890	0.693	0.790
B4+qC	0.715	0.738	0.581	0.678
B4+qD	0.735	0.709	0.644	0.696
B4+qP	0.779	0.796	0.648	0.741
B4+qT	0.780	0.868	0.725	0.791
B4+qF+qW+qT	0.816	0.906	0.743	0.822
ES-all	**0.860**	**0.915**	**0.788**	**0.854**

Table 5: Ablation study of the web (w), query-log (q) and table (t) features (bold letters indicate whole feature families).

Webcrawl term frequencies and document frequencies (from the wF set) are also important. Very frequent and infrequent instances were found to be often incorrect (e.g., respectively *'song'* and *'Brad Pitttt'*). Table PMI (a feature in the qC family) also ranked high in importance: instances that co-occurr very frequently in the same column/row with seeds $\mathcal{S}$ are often found to be correct (e.g., a column containing the seeds *'Brad Pitt'* and *'Tom Hanks'* will likely contains other actors). Other termness (T), frequency-based (F) and co-occurrence (C) features also play some role in the model.

Variable importance is only an intrinsic indicator of feature relevance. In order to better assess the actual impact of the single features on AP, we ran our system on each feature type: results for the web (w), query log (q) and table (t) families are reported in Table 5. For reason of space constraints, we here only focus on some high level observations. The set of web termness features (wT) and frequency features (wF) are alone able to provide a large improvement over *B4* (row 1), while their combination (row 2) does not improve much over the features taken individually.

Seed instances for KE_{dis}

Actors	Athletes			Musicians
Jodie Foster	Bob Gibson	Jared Allen	Randy Moss	Rise Against the Machine
Humphrey Bogart	Don Drysdale	Andres Romero	Peyton Manning	Pink Floyd
Anthony Hopkins	Albert Pujols	Kenny Perry	Jerry Rice	Spice Girls
Katharine Hepburn	Yogi Berra	Martin Kaymer	Robert Karlsson	Pussycat Dolls
Christopher Walken	Dejan Bodiroga	Alexander Ovechkin	Gheorghe Hagi	The Beatles
Gene Hackman	Allen Iverson	Shea Weber	Marco Van Basten	Iron Maiden
Diane Keaton	Yao Ming	Patrick Roy	Zinedine Zidane	John Lennon
Edward Norton	Tim Duncan	Alexei Kovalev	Roberto Baggio	Frank Sinatra
Robert Duvall				Led Zeppelin
Hilary Swank				Freddie Mercury

Seed instances for KE_{pat}

Actors	Athletes	Musicians
Dennis Hopper - The Good Life	Dallas cowboys - Julius Crosslin	Kevin Brown - Corndaddy
Tom Hanks - The Terminal	New york Giants - Plaxico Burress	Barry Gibb - The Bee Gees
Julia Roberts - Mona Lisa Smile	Philadelphia Eagles - Danny Amendola	Patty Smyth - Scandal
Kevin Bacon - Footloose	Washington Redskins - Rock Cartwright	Dave Matthews - Dave Mathews Band
Keanu Reeves - The Lake House	New England Patriots - Laurence Maroney	Gwen Stefani - No Doubt
Marlon Brando - Don Jaun Demarco	Buffalo Bills - Xavier Omon	George Michael - Wham
Morgan Freeman - The Shawshank Redemption	Miami Dolphins - Ernest Wilford	Mark Knopfler - Dire Straits
Nicole Kidman - Eyes Wide Shut	New York Jets - Chansi Stuckey	Brian Jones - The Rolling Stones
Al Pacino - The Godfather		Pete Shelley - Buzzcocks
Johnny Depp - Chocolat		
Halle Berry - Monster's Ball		

10-best ranked instances in one test fold

Actors		Athletes		Musicians	
Gordon Tootoosis	Ron Randell	Rumeal Robinson	Todd Warriner	Colin Newman	Wu-tang Clan
Rosalind Chao	Alimi Ballard	Jeff Mcinnis	Hong-chih Kuo	Ghost Circus	Tristan Prettyman
John Hawkes	Fernando Lamas	Ahmad Nivins	Leon Clarke	Ray Dorset	Top Cats
Jeffrey Dean Morgan	Bruno Cremer	Carlos Marchena	Josh Dollard	Plastic Tree	*Roseanne
George Macready	Muhammad Bakri	Chad Kreuter	Robbie Alomar	*Doomwatch	John Moen

Table 6: Listing of all seeds used for KE_{dis} and KE_{pat}, as well as the top-10 entities discovered by *ES-all* on one of our test folds.

This suggests that wT and wF capture very similar information: they are indeed highly correlated ($r = 0.80$). Rows 5-7 refer to web table features: the features tC outperform and subsume the frequency features tF ($r = 0.92$). For query log features (rows 8-14), only qF, qP and qT significantly increase performance. Distributional and co-occurrence features (qD and qC) have very low effect, as they are mostly subsumed by the others. The combination of qF, qP and qT (row 14) performs as well as the whole $\boldsymbol{q}$ (row 8).

Experiment conclusions: From our experiments, we can draw the following conclusions:

1. Wikipedia features taken alone outperform the baselines, however, web and query log features, if available, subsume Wikipedia features;

2. q, t and w are all important, and should be used in combination, as they drive mostly independent information;

3. the syntagmatic and paradigmatic information conveyed by the two extractors are most relevant, and can be significantly boosted by adding frequency- and termness-based features from other sources.

5 Conclusions and Future Work

In this paper, we presented a general information extraction framework, called Ensemble Semantics, for combining multiple sources of knowledge, and we instantiated the framework to build a novel ML-based entity extraction system. The system significantly outperforms state-of-the-art ones by up to 22% in mean average precision. We provided an in-depth analysis of the impact of our proposed 402 features, their feature families (Web documents, HTML tables, query logs, and Wikipedia), and feature types.

There is ample directions for future work. On entity extraction, exploring more knowledge extractors from different sources (such as the HTML tables and query log sources used for our features) is promising. Other feature types may potentially capture other aspects of the semantics of entities, such as WordNet and search engine click logs. For the ranking system, semi- or weakly-supervised algorithms may provide competing performance to our model with reduced manual labor. Finally, there are many opportunities for applying the general Ensemble Semantics framework to other information extraction tasks such as fact extraction and event extraction.

References

Steven Abney. 1991. Learning taxonomic relations from heterogeneous sources of evidence. In *Principle-Based Parsing*. Kluwer Academic Publishers.

Eric Brill. 1995. Transformation-based error-driven learning and natural language processing: A case study in part of speech tagging. *Computational Linguistics*, 21(4).

Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni. 2005. KnowItNow: Fast, scalable information extraction from the web. In *Proceedings of EMNLP-2005*.

Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *Proceedings of KDD-08*, pages 875–883.

Surajit Chaudhuri, Venkatesh Ganti, and Dong Xin. 2009. Exploiting web search to generate synonyms for entities. In *Proceedings of WWW-09*, pages 151–160.

Philipp Cimiano and Steffen Staab. 2004. Learning by googling. *SIGKDD Explorations*, 6(2):24–34.

Philipp Cimiano and Johanna Volker. 2005. Towards large-scale, open-domain and ontology-based named entity classification. In *Proceedings of RANLP-2005*, pages 166–172.

Philipp Cimiano, Aleksander Pivk, Lars Schmidt-Thieme, and Steffen Staab. 2005. Learning taxonomic relations from heterogeneous sources of evidence. In *Ontology Learning from Text: Methods, Evaluation and Applications*, pages 59–73. IOS Press.

Michael Fleischman and Eduard Hovy. 2002. Classification of named entities. In *Proceedings of COLING 2002*.

Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232.

Jerome H. Friedman. 2006. Recent advances in predictive (machine) learning. *Journal of Classification*, 23(2):175–197.

Zellig Harris. 1964. Distributional structure. In Jerrold J. Katz and Jerry A. Fodor, editors, *The Philosophy of Linguistics*, New York. Oxford University Press.

Marti A. Hearst. 1992. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of COLING-92*, pages 539–545.

Jian Hu, Gang Wang, Fred Lochovsky, Jian tao Sun, and Zheng Chen. 2009. Understanding user's query intent with Wikipedia. In *Proceedings of WWW-09*, pages 471–480.

Dekang Lin. 1998. Automatic retrieval and clustering of similar words. In *Proceedings of COLING-ACL-98*.

Joseph F. Mc Carthy and Wendy G Lehnert. 2005. Using decision trees for coreference resolution. In *Proceedings of IJCAI-1995*, pages 1050–1055.

Shachar Mirkin, Ido Dagan, and Maayan Geffet. 2006. Integrating pattern-based and distributional similarity methods for lexical entailment acquisition. In *Proceedings of ACL/COLING-06*, pages 579–586.

Marius Paşca and Benjamin Van Durme. 2008. Weakly-supervised acquisition of open-domain classes and class attributes from web documents and query logs. In *Proceedings of ACL-08*, pages 19–27.

Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. 2006. Organizing and searching the world wide web of facts - step one: The one-million fact extraction challenge. In *Proceedings of AAAI-06*, pages 1400–1405.

Marius Paşca. 2007. Weakly-supervised discovery of named entities using web search queries. In *Proceedings of CIKM-07*, pages 683–690, New York, NY, USA.

Patrick Pantel and Marco Pennacchiotti. 2006. Espresso: A Bootstrapping Algorithm for Automatically Harvesting Semantic Relations. In *Proceedings of ACL-2006*, Sydney, Australia.

Patrick Pantel, Eric Crestan, Arkady Borkovsky, Ana-Maria Popescu, and Vishnu Vyas. 2009. Web-scale distributional similarity and entity set expansion. In *Proceedings of EMNLP-09*, Singapore.

Ellen Riloff and Rosie Jones. 1999. Learning dictionaries for information extraction by multi-level bootstrapping. In *Proceedings of AAAI-99*, pages 474–479.

Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. 2008. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of EMNLP-08*, pages 582–590.

Bin Tan and Fuchun Peng. 2006. Unsupervised query segmentation using generative language models and wikipedia. In *Proceedings of WWW-06*, pages 1400–1405.

Hristo Tanev and Bernardo Magnini. 2006. Weakly supervised approaches for ontology population. In *Proceedings of EACL-2006*.