Exploring the Relationship of Information Retrieval and Information Extraction

Stefan Schoenmackers

September 15, 2006

Abstract

Information Retrieval (IR), Information Extraction (IE), and Text Classification (TC) are typically seen as related, but distinct fields. In IR the goal is to return documents relevant to a query, in IE the goal is to return strings representing instances of a particular class or relation, and in TC the goal is to determine to which class a test document belongs. While there are many techniques for each of IR, IE, and TC, we argue that for a reasonable and effective subset, the main distinction reduces to how the query and corpus are represented. We show how these subsets of IR, IE, and TC can be transformed to each other, and use these transformations to explore the implicit assumptions, limitations, and relationship between example IR, IE, and TC systems. Finally, we use these insights to build a hybrid IE system, which uses IR distance metrics to improve the IE system's results.

1 Introduction

Information Retrieval (IR), Information Extraction (IE), and Linear Text Classification (TC) are typically seen as related, but distinct fields. These fields are related since they all deal with textual information, but distinct since they have separate goals. IR systems are designed to return a ranked list of documents related to a user's query. IE systems are designed to return specific entities, properties, and relationships of interest to the user. TC systems are designed to determine the most likely class for a document. For example, if a user was interested in "Presidents of the United States", an IR system would return documents related to those keywords, in which they could (hopefully) find all the U.S. presidents. Similarly, a TC system would (hopefully) determine which particular documents in a corpus mention U.S. presidents. An IE system, on the other hand, would (hopefully) return a list of strings containing "George Washington", "Abraham Lincoln", etc.

Although there is a wide variation in methods, some techniques in each field have proven popular and effective. IR systems based on the 'document vector' model are effective and fairly well understood, TC methods such as Naive Bayes or other linear models generally perform well despite their simplicity, and IE systems that extract items based on pre-specified and/or learned patterns have been effective in a number of IE tasks. In particular, recent research into information extraction systems (e.g. [8] [3] [30] [28]) has shown that using fairly simple pattern-based IE

techniques and simple linear combinations of evidence across patterns and documents can provide efficient, effective, and scalable information extraction systems. In this document we focus such IR, IE, and TC systems.

From a high level, IR, IE, and TC are all performing a form of classification based on contexts, using both local and global corpus statistics to do so. Some research in IR, IE, and TC focuses on making a binary decision (is a doc relevant or not, is an extraction correct or not, or does a document belong to this class or not). However, many techniques in each field also include some estimate of the confidence of the decision, allowing an easy way to tune these systems. We take this more general approach by viewing IR, IE, and TC systems as ranking documents, extractions, or classes respectively, and explore the assumptions of and relations between their scoring techniques.

In this document we seek to understand and codify the relationship between the classification/ranking metrics used by the previously mentioned document vector-based Information Retrieval systems, pattern-based Information Extraction systems, and Linear Text Classification systems. While the relationship between IR and Naive Bayes text classification is well understood, we extend it to include their relationship with pattern-based IE methods. We give simple transformations for converting between problems in IR, IE, and TC, and argue that they are all different aspects of the same ranking problem, merely representing the corpus in different ways. We further use these transformations to explore the inherent assumptions and relationship between example IE and IR systems. Finally, we propose and evaluate a hybrid IE system which performs extraction based on an underlying IE system, but uses IR term weighting and distance metrics to improve its classification/ranking of extractions.

In this document we do not focus on where the corpus, query, patterns, etc. originate, but rather treat them in abstract terms and focus on how these IR, IE, and TC systems compute confidences. We also assume that the IE segmentation problem (determining which words need to be extracted as part of the entity, property, or relation, and which words should be left out) is solved externally. This is a reasonable assumption since many effective pattern-based IE systems rely on simple heuristics (e.g. [30]), part of speech taggers (e.g. [8]), named entity taggers (e.g. [3]), or other natural language processing tools to perform the segmentation. Finally, although all the systems we consider rely on a linear combination of evidence, this still covers many interesting and effective systems. Restricting our study to linear IR, IE, and TC systems allows us to more easily discuss their relationships and inherent assumptions.

The main contributions of this paper are as follows:

- We give simple transformations which allow us to convert IE problems or TC problems to IR ones, and vice versa.
- We analyze the assumptions under which a typical IE and IR system produce equivalent rankings, and leverage that to elucidate the assumptions of each system and the relationship between them.
- We define a hybrid IE system, which leverages IR term weighting functions to improve the precision/recall trade-off of a modern IE system.

2 Background

2.1 Information Retrieval

The goal of Information Retrieval (IR) systems is to return a ranked list of documents relevant to a query. While there are a wide variety of techniques, a common and effective method for doing this is based on the 'document vector' model. In this model, each document is represented as a $|V|$ dimensional vector (where $V = \{w_1, w_2, ..., w_n\}$ is the set of words in the corpus) whose ith entry is, for example, the number of times word w_i appears in the document. Queries are similarly represented as $|V|$ dimensional vectors, whose ith entry represents some notion of how relevant word i is to the query. Individual document scores are computed as a vector distance between the query and document vector, and the overall query results are documents ranked by this score. For example, one scoring method is the tf-idf metric defined as:

Definition 1. *Let $IR_{tf-idf}(C, q, d)$ be a function that takes as input a corpus C, query q, and document d_i and returns a real-valued score for d_i computed as*

$$IR_{tf-idf}(C, q, d_i) = \sum_{j \in V} d_{ij} * q_j * \log \frac{N}{n_j} \tag{1}$$

where V is the set of words in the corpus, q_j is the number of times word j appears in the query, d_{ij} is the number of times word j appears in document d_i, N is the total number of documents in the corpus, and n_j is the total number of documents containing word j. For two documents d_i and d_j, if $IR_{tf-idf}(C, q, d_i) > IR_{tf-idf}(C, q, d_j)$ then d_i is a better match for the query than d_j.

In this case, the document vectors are constructed by simply counting word frequencies, and the query vector is constructed assuming that word j's relevance is $q_j * \log \frac{N}{n_j}$. This word relevance is designed to model the word's prevalence in the query, modified by how rare it is within the corpus.

2.2 Text Classification

In Text Classification (TC), the goal is to predict to which class unseen texts belong. For example, classifying an incoming email as spam/not spam, predicting which newsgroup an article was posted to, or predicting which news articles a user will find interesting. TC is typically viewed as a standard machine learning problem, and as such, there are a wide number of available techniques. Linear techniques such as Naive Bayes and Support Vector Machines have proven quite effective, and so we focus on linear systems in this work.

These linear Text Classification techniques are similar to many IR techniques in that both treat words as independent features. These linear TC techniques differ from standard IR techniques in that they typically use much more training data. IR typically assumes either no training data or a few of the top examples tagged as relevant/not relevant by a user (referred to as 'relevance feedback'.) Rather than focusing on making a decision, Text Classification can be seen as ranking classes based on how well they match a test document.

We now consider a Naive Bayes text classifier as an example TC system. The Naive Bayes model has been shown to work well for a large number of text classification tasks. The model assumes that the probability of seeing a particular word depends only on the class, and is independent of all other words. Even though this word independence assumption clearly does not hold for natural language text, this model has achieved good results in practice. Additionally, although the Naive Bayes model generally tends to produce highly polarized probabilities, it also tends to preserve the relative ordering between classes. As noted in [6], this is sufficient for accurate results.

The basic scoring metric for a Naive Bayes classifier is given in Definition 2. For optimal performance, a document is assigned to the class maximizing this score.

Definition 2. *For a document d and class c_i, the likelihood that d is in class c_i is computed as*

$$LTC_{Naive-Bayes}(d, c_i) = P(c_i) \prod_{1 \leq j \leq |V|} P(w_j|c_i)^{d_j} \tag{2}$$

where $P(c_i)$ is the prior probability of class c_i, $P(w_j|c_i)$ is the probability that word j appears in class i, and d_j is the number of times word j appears in document d. For two classes c_1 and c_2 if $LTC_{Naive-Bayes}(d, c_1) > LTC_{Naive-Bayes}(d, c_2)$ then d is more likely to be a member of class c_1 than c_2.

If we set $P(w_j|c_i) = \frac{n_j+1}{n+|V|}$ where n_j is the number of times word j appears in all documents of class i, and n is the total number of words in all documents of class i, then this is the multinomial Naive Bayes model as given in [16] and [18]. We refer the reader to those for a more in-depth discussion and analysis of Naive Bayes text classifiers.

2.3 Information Extraction

Information Extraction (IE) systems are focused at a much finer granularity than IR or TC. Rather than returning documents, IE systems return strings representing instances of a desired class or relation. While there are a wide variety of techniques of varying effectiveness and complexity, systems that extract based on learned or pre-specified patterns have been shown to be simple and effective. We focus on these pattern based systems.

Generally speaking, a pattern is an alternating sequence of tokens (literal words, phrases, characters, etc.) and slots (placeholders for items to be extracted, possibly including part of speech, type, etc. constraints). For example, in the pattern "cities such as <PN>" the words 'cities', 'such', and 'as' are the tokens, and '<PN>' is the slot, constrained to be a proper noun. A pattern matching a sentence requires two things. Firstly, the sentence must contain substrings matching the pattern tokens, in the same order as in the pattern. Secondly, the substrings matching the tokens must be separated by words/phrases matching the constraints on the slots. When these conditions are met, the tuple of words/phrases matching the slots is extracted. For example, the pattern above would match the sentence 'I like cities such as New York.' and extract 'New York' from it. Different pattern-based IE systems may define the notion of a token differently (e.g. KnowItAll [8] uses string literals whereas Snowball [3] uses more complex term vectors), and may place different constraints on the slots (e.g. KnowItAll uses a simple part of speech tagger whereas Snowball uses a

named entity recognizer.) We assume that the constraints on the slots, as well as the surrounding tokens, handle the segmentation problem. Furthermore, for this work we assume an abstract set of patterns and instead focus on how these systems estimate their confidence in each extraction.

For the early Message Understanding Conference (MUC) tasks, the goal was to extract every occurrence in every document in isolation. However, we focus on more recent systems such as Snowball [3], KnowItAll [8], and NOMEN [30], which have dropped this requirement. Instead, they allow extractions to be missed in particular documents as long as they are found in others. These systems determine the confidence of an extraction by combining evidence across multiple documents.

From a high level, pattern based IE systems operate in two stages: candidate generation and confidence estimation. They first search the corpus for matches to their patterns and create a set of candidate extractions. They then compute the confidence of each extraction based on which patterns matched, how well the patterns matched, and/or how often the patterns matched. While there are some interesting techniques for improving the efficiency of the candidate generation process (e.g. [1]), in this work we are mainly interested in the confidence estimation techniques. As a concrete example, NOMEN and Snowball assume that patterns extract items independently, and so use a Noisy-Or function to compute extraction confidence.

Definition 3. *Let $IE_{noisy-or}(C, r, e)$ be a function that takes as input a corpus C, class or relation r, and candidate extraction e and returns a real-valued score for e computed as*

$$IE_{noisy-or}(C, r, e) = 1 - \prod_{p \in P} (1 - Pr(p))^{n_{pe}} \tag{3}$$

where P is a set of patterns indicative of class/relation r, $Pr(p)$ is an estimate of the probability that pattern p extracts a correct instance, and n_{pe} indicates how many times pattern p extracts e from the corpus.

This scoring technique assumes that the incorrect extractions for one pattern are independent of the incorrect extractions for other patterns. The Noisy-Or function computes the probability that an extraction is correct based on how likely it is that all patterns extracted it incorrectly. NOMEN and Snowball restrict n_{pe} to be a binary value, but this is not a necessary requirement. While the results generated by this method tend to be polarized towards one, in practice they provide a way for ranking the extractions. This allows the system to distinguish more likely extractions from less likely ones, providing a reasonable way to trade precision and recall.

Although it uses a different scoring function, this extraction method is similar in spirit to KnowItAll, AutoSlog [22], and similar systems that use pre-specified patterns. Additionally, KnowItAll observed that domain independent patterns can reliably extract a wide variety of classes and relations, and that the confidence for these patterns remains consistent across a variety of classes. So while this definition may seem to make idealistic assumptions, these assumptions are not unrealistic. In practice, the patterns and corresponding confidences are often learned iteratively based on some seed patterns/examples, but in this work we treat this as an abstract process. We can treat these iterative pattern learning techniques as a pre-processing step, or alternatively, the confidence estimation ideas presented below can be used as part of an iterative, pattern learning process.

In this work we are focusing on pattern-based systems which rely on a linear combination of evidence. We chose to study these types of systems instead of more complex, nonlinear ones using, for example, HMMs [9] or CRFs [17], since these pattern-based systems are simpler, require less training data, and scale to large data sets more efficiently.

3 Exploring the Relationship between IR, IE, and TC

On the surface the techniques for IR, IE, and TC appear to be quite different. However, we now argue that they are fundamentally similar – they just model the corpus and query in different ways.

From a high level, they all attempt to use contextual information to estimate the confidence that each item is correct. Specifically, these IR systems use query, document, and corpus word counts to return more relevant documents before less relevant ones, these IE systems use which, how often, and how well patterns match extractions to distinguish more likely extractions from less likely ones, and these TC systems use document, class, and corpus word counts to determine how likely each class is for a test document. While IE and TC systems ultimately decide whether an extraction is correct or which class the document belongs to, by ranking items based on their confidences, we provide a much finer-grained control for this decision.

In the following sections, we formalize this high level similarity by showing that these subsets of IR, IE, and TC are all computationally equivalent. To do so, we show a set of simple transformations which convert a problem X in IR, IE, or TC to a problem Y in either of the other two, such that solving Y produces results equivalent to those of X.

3.1 Transforming IR problems to IE Problems

We first show that, through a simple transformation of the corpus and query, a pattern-based IE system can generate the same ranking of documents as an IR system.

The main idea behind this is to transform the IR notions of a corpus, vocabulary, and query into the IE notions of a corpus, patterns, and pattern confidences. To do so, we build an extractable term corresponding to each document in the corpus. We then construct a new corpus and populate it with 'sentences' of the form '$w\ doc.id$', for each word w in every document in the original corpus. Finally, to retrieve the relevant documents we extract document ids from this new corpus, using '$queryterm_i$ <X>' as the patterns. Under this transformation, the IR notion of a document becomes analogous to the IE notion of a potential extraction, and the IR notion of a keyword becomes analogous to the IE notion of an extraction pattern.

Algorithm 1 lists pseudocode for this transformation. Note that although it uses a particular pattern confidence function on line 23 and a particular IE scoring function on line 4, alternatives may be used without affecting the basic transformation.

One interesting application of this transformation is that it allows us to explore the nature and inherent assumptions these algorithms make. For example, by determining assumptions such that $IE_{noisy-or}$ ranks documents in the same order as IR, we gain some insight on what IR_{tf-idf} is computing and how it works. Using the reduction in Algorithm 1, including the particular pattern

Algorithm 1 Pseudocode transforming an IR problem into an IE problem

1: **function** COMPUTEIRASIE(C:corpus, q:query, d:document)
2: $C' \leftarrow$ ConvertIRCorpus(C)
3: P $\leftarrow$ ConvertQueryToPatterns(C,q)
4: **return** $IE_{noisy-or}(C'$, P, d)
5: **end function**

6: **function** CONVERTIRCORPUS(C)
7: $C' \leftarrow \emptyset$
8: **for all** $doc \in$ C **do**
9: $doc' \leftarrow \emptyset$
10: **for all** $word \in doc$ **do**
11: $newSentence \leftarrow$ '$word\ doc.id$.'
12: $doc' \leftarrow doc' \bigcup \{newSentence\}$
13: **end for**
14: $C' \leftarrow C' \bigcup \{doc'\}$
15: **end for**
16: **return** C'
17: **end function**

18: **function** CONVERTQUERYTOPATTERNS(C,q)
19: $P \leftarrow \emptyset$
20: **for all** $word \in$ q **do**
21: $p_i \leftarrow$ '$word$ <X> .'
22: $P \leftarrow P \bigcup \{p_i\}$
23: $Pr(p_i) \leftarrow \frac{N^{q_i} - n_i^{q_i}}{N^{q_i}}$
24: **end for**
25: **return** P
26: **end function**

confidence on line 23, we can show that $IE_{noisy-or}$ ranks documents in the same order as IR_{tf-idf}. More formally:

Theorem 3.1. *For a given IR corpus C, query Q, and documents d_1 and d_2, if C' and P are respectively the corresponding IE corpus and set of patterns generated by Algorithm 1, and if we estimate the pattern confidence as $Pr(p_i) = \frac{N^{q_i} - n_i^{q_i}}{N^{q_i}}$, then*

$$IR_{tf-idf}(C, Q, d_1) > IR_{tf-idf}(C, Q, d_2) \Leftrightarrow IE_{noisy-or}(C', P, d_1.id) > IE_{noisy-or}(C', P, d_2.id) \tag{4}$$

The proof results from substituting the terms into Equation 1. It is listed in full in Appendix A.

From the proof we can make an interesting observation. We can see that the tf-idf IR system from Definition 1 effectively ranks the documents in a Noisy-Or fashion, estimating the individual word probabilities as a specific function of how prevalent the word is in the corpus and the query.

Again, this reduction is not limited to just showing that IR_{tf-idf} can be reduced to $IE_{noisy-or}$. By weighting the pattern confidences on line 23 of Algorithm 1 differently, we can reduce a variety of IR scoring functions to a variety of IE ones. However, not all such reductions are possible; the IE pattern confidence function may limit which IR systems can be reduced to it. For example, using a Noisy-Or combination of probabilities as in $IE_{noisy-or}$ causes the individual pattern confidences to be weighted as $\log \frac{1}{1-Pr(p_j)}$ (equation 6 in Appendix A). However, since $Pr(p_i) \in [0, 1]$, all of these weights are constrained to be non-negative. While this restriction holds for IR_{tf-idf} (since $q_i \log \frac{N}{n_i} \geq 0$), it is not guaranteed for all IR systems.

So although this reduction is more general, for an arbitrary IR system to be reduced to and solved by an IE system, the IE evidence combination function must be expressive enough to allow it. Reductions where this is not possible describe, at least in a weak way, some properties and limitations of the corresponding IE or IR systems. Thus, while the Noisy-Or evidence combination method has been shown to work well, it is not as general as it could be, since it effectively includes no negative evidence.

3.2 Transforming IE problems to IR Problems

We now show a transformation which allows us to perform the classification/confidence estimation step of IE using IR methods. Specifically, we show how an IR system can be used to rank potential extractions by decreasing confidences.

The main idea mirrors the transformation in Section 3.1. Instead of creating an extractable term for every document, we create a new document for every potential extraction. We populate each extraction's document using the patterns it appears with, generate a query vector weighted according to pattern confidence, and score each extraction based on how well its corresponding document matched the query. Algorithm 2 lists pseudocode for this. Again, although it uses a particular query weighting function on line 4 and a particular IR scoring function on line 23, alternatives may be used without affecting the basic transformation.

This transformation has the IE patterns P play the role of the vocabulary V for an IR system, and the pattern confidences dictate the query terms and weights. Thus, it effectively reverses the ideas and operations from the previous transformation.

As before, this transformation allows us to explore the nature and inherent assumptions of these algorithms. Using the particular reduction and weighting scheme as given in Algorithm 2, we can show that the IR_{tf-idf} function will order extractions in decreasing order of the confidence computed by $IE_{noisy-or}$. More formally:

Theorem 3.2. *For a given corpus C, set of patterns and corresponding confidences P, and extractions ex_1 and ex_2, if we construct the corpus C' and query Q with term weights as in Algorithm 2, and assume that each pattern matches at least one extraction, then*

$$IE_{noisy-or}(C, P, ex_1) > IE_{noisy-or}(C, P, ex_2) \Leftrightarrow IR_{tf-idf}(C', Q, d_1) > IR_{tf-idf}(C', Q, d_2)$$

where d_1 and d_2 are the constructed documents corresponding to ex_1 and ex_2 respectively.

The proof is similar to that of Theorem 3.1. It follows from simple substitutions into equations 3 and 1, and is given fully in Appendix B.

Algorithm 2 Pseudocode transforming an IE problem into an IR problem

function COMPUTEIEASIR(C:corpus, P:patterns, e:extraction)
    C'=ConvertIECorpus(C, P)
    Q=ConstructQuery(P, C')
    return IR_{tf-idf}(C', Q, e)
end function

function CONVERTIECORPUS(C, P)
    $C' \leftarrow \emptyset$, $V \leftarrow$ P ▷ Let P define the 'words' (vocabulary) of C'
    for all Sentences $s \in$ C **do**
        for all $p_i \in P$ **do**
            if p_i matches s and extracts e **then**
                Append p_i to the document named e (creating and adding it to C' if necessary)
            end if
        end for
    end for
    return C'
end function

function CONSTRUCTQUERY(P, C')
    $Q \leftarrow \emptyset$
    for all $p_i \in P$ **do**
        Let E be the number of distinct extractions
        Let e_i be the number of distinct extractions matching p_i
        $Q \leftarrow Q \bigcup \{p_i\}$
        $q_i \leftarrow \frac{\log(1-Pr(p_i))}{\log \frac{e_i}{E}}$
    end for
    return Q
end function

It is important to note that, although this transformation appears to require a specific set of patterns, it can easily consider *all* potential patterns by simply choosing pattern confidences which only select the relevant ones. For example, starting with a set of patterns P, we can easily expand this set to P' by including all sequences of up to k words. Then simply setting $Pr(p_i) = 0$ for all patterns $p_i \notin P$ would produce identical results. We used a pre-specified set of patterns since it captures the important aspects of the reduction and simplifies the notation.

3.3 Transforming between TC problems and IR problems

The relationship between Information Retrieval and Text Classification is fairly well understood. For example, [15] and [13] discuss and analyze some aspects of their similarity. In the interest of conserving space, we informally sketch a transformation for converting IR to TC and vice versa,

and rely on those sketches and the previous research to describe their relationship.

To transform a TC problem to an IR one, we create a new corpus consisting of one document per class. These class documents are created by simply concatenating all examples of the class. Classifying a test document can be performed by using appropriate word weights and looking at the order in which those 'documents' are retrieved. The most top document returned will correspond to the most likely class, the next to the second most likely class, etc.

To transform an IR problem to a TC one, we can view each document in the corpus as defining a class. By weighing the terms appropriately, creating a 'test document' q from the query, and ordering the newly created 'classes' by how likely q is to belong to each, we can compute results equivalent to an IR system.

Just as before, limitations on the set of legal weights for particular IE or TC systems limit which can be reduced to each other. For example, the IR_{tf-idf} weighting scheme cannot be reduced to $LTC_{Naive-Bayes}$, since all the IR_{tf-idf} terms take the form: $\log \frac{N}{n_i} \geq 0$, whereas all the $LTC_{Naive-Bayes}$ terms take the form: $\log P(w_j|c_i) \leq 0$.

However, as an interesting side note, if we instead consider the compliment version of Naive Bayes (i.e. finding $argminP(\overline{c}_i)$ instead of $argmaxP(c_i)$), then there are probabilities such that IR_{tf-idf} can be reduced to $LTC_{Naive-Bayes}$ and vice versa. Namely, if we assume $P(w_j|\overline{c}_i) = \frac{n_j^{c_{ij}}}{N^{c_{ij}}}$ where c_{ij} is the number of times word j appears in class i (document i, from the transformation outlined above), then we see computing the maximum tf-idf score for a document is equivalent to computing the minimum compliment class. The proof is similar to that of Theorem 3.1, and is left as an exercise to the reader.

Thus, in a way, the given tf-idf metric is finding the documents that are not unlikely, rather than the documents that are likely. This similar to an observation in [12], which demonstrated that the compliment version of a Naive Bayes text classifier can significantly outperform the standard version. While the probability estimates here are different, they behave similarly. Namely, the rarer the word, and the more frequently it occurs in a class (document in the IR case), the more likely it makes that class (document). We believe that the relationship between IR_{tf-idf} and the compliment $LTC_{Naive-Bayes}$ and the surprisingly good performance of both is an interesting correlation.

3.4 Comparing the Systems

Given the transformations above, we see that IR, IE, and TC are all just different representations of the corpus. Using these transformations we see that the IR notion of a document, the TC notion of a class, and the IE notion of an extraction are all analogous. Similarly, we see that the contexts – the set of IE patterns P or the vocabulary V of IR and TC – are also analogous.

Thus, we observe the subset of IR, IE, and TC systems that we are concerned with vary along four dimensions:

1. Textual Objects - what is being returned? Is it Documents (for IR), Classes (for TC), or Extractions (for IE).

2. Contexts - what is the basic set of context (vocabulary) for the textual objects? How many

are there, and how long are they? Typically, it is a large number words or a much smaller number of patterns, but it may contain additional information such as part of speech tags, named entity tags, etc.

3. Object Context Weights - For a given textual object, how related is each context?

4. Context Confidence Weights - How much does each context indicate the desired query/class/relation?

Of note is that individual techniques within Information Retrieval, Linear Text Classification, or pattern-based Information Extraction are simply variations along all but the first of these same dimensions. For example, IR methods can use some prior knowledge (relevance feedback techniques), there are many variations on what vocabulary to use (eliminate stopwords, stem words, etc.), and there are a wide number of techniques for weighting the document (context) and query vectors. The transformations above show that subsets of IR, IE, and TC are related, and simply variations along on one additional dimension.

Finally we observe that the transformations above not only allow us to explore the relationship between IR, IE, and TC, they also allow us to pick and choose aspects from each. For example, using the reduction defined in Algorithm 2 we can build a hybrid IE/IR system that extracts items based on patterns, but ranks the extractions based on IR similarity metrics. Since IR weighting techniques can account for data sparsity, we believe they can help with data sparsity problems in IE. In the next section we explore some of these possibilities.

4 Experiments

4.1 Setup

Having explored the relationship between these subsets of IR, IE, and TC, we now seek ways of leveraging this insight. In this section we demonstrate that using term weighting techniques from IR can improve the precision/recall trade off of the Named Entity Recognition and Relation Extraction tasks of an existing IE system.

As motivation, we first observe that many of these pattern-based IE systems (e.g. [8], [3], [23]) rely on a relatively small number of high-precision patterns (i.e. $|P| << |V|$). While these patterns tend to extract correct items, they also tend to extract a large number of items only once or twice. For these rare items it is difficult to verify whether they are correct but uncommon, or whether they were extracted due to syntactic anomalies. Yet by only matching based on high-precision patterns, these methods implicitly discard a large amount of useful data: all other contextual information about the extraction. While these additional contexts may not reliably extract items, they capture how the extraction "behaves" in the corpus. We can use these contexts to raise or lower the confidence in an extraction based on how similar each extraction is to others.

The problem of determining similarity based on weak contextual information has long been studied in Information Retrieval. IR systems have effective techniques to manage and model a large amount of loosely related data (i.e. the words in a document). IR techniques for combining evidence from across words, across documents, and from relevance feedback mechanisms are

fairly well understood. Additionally, these techniques have been shown to scale quite well to larger vocabularies and larger numbers of documents. So rather than concocting a notion of similarity, we leverage our observations and the reduction from Section 3.2 to use IR scoring techniques to compute the extraction's confidence.

We implemented a hybrid IE system that extracts items using an (arbitrary) underlying IE system, but ranks them using IR term-weighting techniques over their full set of contexts. For these experiments we use KnowItAll [8] as the underlying IE system, but ideas easily apply to other IE systems.

We tested hybrid systems based on two common IR weighting functions: the simple tf-idf function used in Definition 1 and Okapi's $BM25$ term weighting function, which was introduced by Robertson et. al. [25] in TREC-3 and has been quite successful since. We denote our hybrid systems using these algorithms as IE_{tf-idf} and IE_{BM25} respectively. The $BM25$ similarity metric is defined as

$$BM25(d_i) = \sum_{j \in Q} \frac{(k_1+1) * d_{ij}}{k_1((1-b)+b*dl/adl)+d_{ij}} \frac{(k_3+1)*q_j}{k_3+q_j} \log \frac{(r+0.5)/(R-r+0.5)}{(n-r+0.5)/(N-n-R+r+0.5)} \tag{5}$$

Where N is the number of documents in the corpus, n is the number containing term j, R is the number of documents known to be relevant, r is the number of relevant documents containing j, d_{ij} and q_i are the frequency of term j in the document and query respectively, dl is the document length, adl is the average document length, and k_1, b, and k_3 are tuning parameters. For all experiments we set $k_1 = 3$, $b = 1$, and $k_3 = 0$.

We adapt this weighting scheme directly to our problem by using the reduction described in Section 3.2. Namely, we build a suitable corpus by creating a 'document' for each extraction using a 'vocabulary' of all contexts of the extraction. We construct the "query" documents by borrowing the IR notion of pseudo-relevance feedback; we assume the top (most frequently extracted) extractions are correct and simply aggregate them. For the system based on IR_{tf-idf}, d_{ij} is the context count for the test extraction, and q_j is the number of times the context appeared with the top extractions. Since we assume the top extractions are correct, we use them, just as many IR systems do, to determine R and r for the IE_{BM25} hybrid system.

There are two important features of these hybrid systems that should be noted. First of all, using this pseudo-relevance feedback technique requires no additional hand-tagged data, so it maintains the unsupervised/self-supervised nature of the underlying IE system (if present). Secondly, since the hybrid systems are based on these IR techniques they do not require any negative examples, but instead leverage corpus statistics.

We tested this hybrid system on two typical IE tasks: Named Entity Recognition and Relation Extraction.

4.2 Named Entity Recognition

Named Entity Recognition (NER) is the task of finding strings specifying instances of a particular class (e.g. Cities, Astronauts, Fruits, etc.). KnowItAll [8] performs named entity recognition by

searching the web using a set of domain-independent hyponym patterns identified by Hearst[11] and evaluating them using corpus statistics. For example, to find instances of the class City, it first queries a search engine to find websites containing the hyponym patterns (e.g. "cities such as <X>", "<X> and other cities", etc.) After downloading these web pages and finding the relevant sentences, it uses a simple part-of-speech tagger to segment the extraction.

The principle technique KnowItAll uses to evaluate extractions is based on the Turney's PMI-IR algorithm [29]. To evaluate a particular extraction e, KnowItAll uses search engine statistics to compute the Pointwise Mutual Information(PMI) between e and every extraction pattern $p_i \in P$. It calculates the PMI as $PMI(p_i, e) = \frac{|Hits(p_i+e)|}{|Hits(e)|}$, where $Hits(p_i + e)$ is the number of web pages containing both the extraction and the pattern and $Hits(e)$ is the number of web pages containing the extraction e. KnowItAll automatically finds thresholds for these PMI scores, uses them as features in a Naive Bayes classifier, and uses that classifier to compute the final confidence in the extraction. See [8] for a more detailed description. We refer to this as 'Baseline-PMI' in the experiments. For these NER experiments, we also use a second, simpler baseline inspired by [4]. To compute confidence, it assumes all extractions are correct, and estimates the confidence of e as the total number of times e was extracted. We refer to this metric as 'Baseline-Num Extr' in the experiments.

For these experiments we tested our system on three classes: Cities, Countries, and Films. For each class, we made a test set by sampling 1000 extractions from a previous run of KnowItAll and hand tagging them as correct/incorrect. We used the sentences KnowItAll found matching each pattern, and we simulated having a larger corpus/set of contexts by adding all sentences mentioning an extraction from the top 50 web pages for each extraction.[1] We ignored case and replaced a few basic types (e.g. numbers) with canonical representations, but otherwise performed no word stemming or other modifications to the sentences. This yielded about 350 sentences per extraction.

We defined the set of contexts as any three words around an extraction. I.e. we use a 4-gram model for the contexts of each extraction. For example, the sentence "I love Paris in the springtime" would produce the contexts 'I love <X> in', 'love <X> in the', and '<X> in the spring' for Paris. In total there were 1,054,668 distinct contexts, and an average of 931.8 total contexts/extraction. For our hybrid system, we used the 10 most frequently extracted items for each class as the pseudo-relevance feedback items.

Figures 1, 2, and 3 show the precision-recall curves for each of the three classes. Using either of these IR techniques to leverage the extra contextual information clearly helps, especially for the Film and City classes. For the Film case in particular, both IE_{tf-idf} and IE_{BM25} nearly double the recall at 90% precision, or cut the error rate by 2/3 at 50% recall. The improvements are less well defined for extracting Countries, but this is unsurprising since the total set of countries is so small, and many of the errors 'behave' like countries textually (e.g. 'Puerto Rico', 'USSR', or 'Members of the European Union').

[1] This uses the same number of search engine queries as computing the PMI between an extraction and four patterns. Thus, we are not using more queries than the baseline KnowItAll system, so our proposed method is no less efficient than KnowItAll in terms of search engine queries. This is important, since search engine query limits were the bottleneck in KnowItAll.

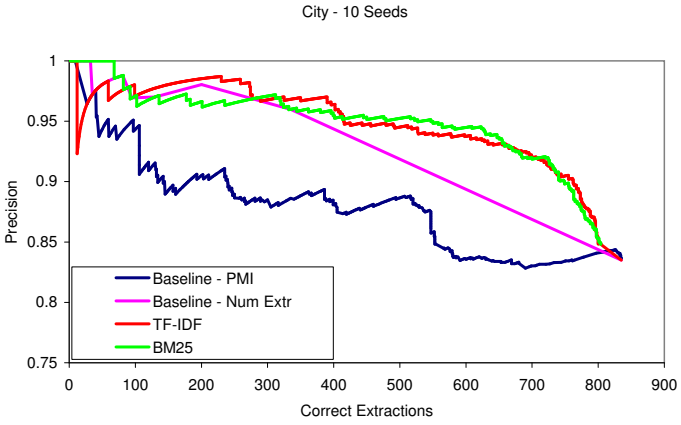

Figure 1: P-R curve for extracting Cities

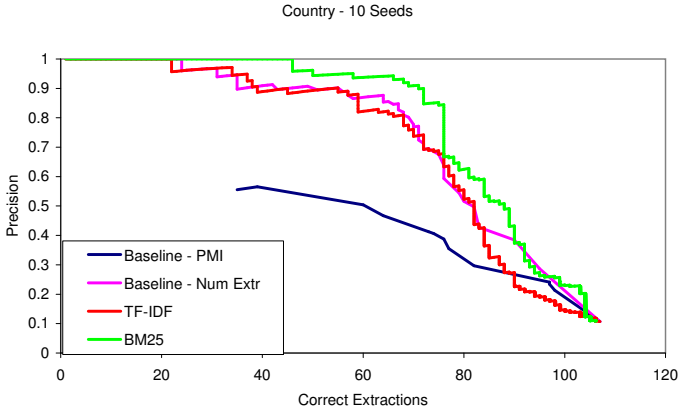

Figure 2: P-R curve for extracting Countries

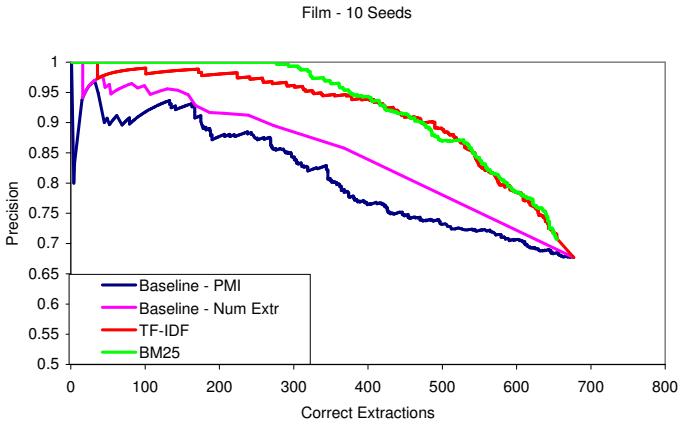

Figure 3: P-R curve for extracting Films

There are three main reasons for the improved performance of IE_{tf-idf} and IE_{BM25}. Firstly, in the Film and City classes there are a large number of items extracted only once. For these rare extractions, any additional contextual information helps determine how city-like/film-like they are, which allows the system to increase/decrease confidence in the extraction appropriately.

Secondly, the additional data helps detect type errors. For example, given the phrase "Films such as George Lucas's epic masterpiece 'Star Wars' ", the KnowItAll patterns will extract 'George Lucas' as the name of a film. The additional contexts of 'George Lucas' will demonstrate that he is not very movie-like, and will thus decrease the system's confidence that he is a film.

Finally, the third source of improvements is in detecting segmentation errors. KnowItAll determines the boundaries of the extraction based on a part-of-speech tagger. While this typically works well, with film names like 'Dumb and Dumber' or 'So I Married an Axe Murderer', it incorrectly extracts 'Dumb' and 'So I Married'. For these extraction errors, their contexts include a large number of occurrences of '<X> and Dumber ...' or '<X> an Axe Murderer', which are not typical contexts for movie titles. Thus, their collections of contexts looks un-film-like, correctly causing both hybrid systems to decrease those confidences.

4.3 Relation Extraction

Relation Extraction (RE) is similar to Named Entity Recognition, but instead of extracting a single instance of a class, its goal is to extract two or more entities that have a particular relationship to each other. For example, we may wish to find mayors with their corresponding cities, or discover which companies acquired other companies. For pattern-based IE systems, the main difference between RE and NER is that RE has more than one slot in the pattern.

For these experiments we tested four relations: MayorOf(Mayor, City), CeoOf(CEO, Company), Acquired(Company, Company), and MergedWith(Company, Company). For the MayorOf and CeoOf relations, their first argument is bound to one of 100 known cities/companies. To provide an interesting mix, 50 of the cities/companies were fairly common ($> 100,000$ search engine hits) and 50 were fairly rare ($< 10,000$ hits). The Acquired and MergedWith relations had neither argument bound.

For these experiments we performed a better head-to-head comparison between the baseline and hybrid systems by forcing them to use the same corpus of sentences. We built this corpus by searching the web for sentences containing the name of the relation (and one of the bound arguments where applicable.) We ended up with 15,355 sentences containing 'ceo' or 'chief executive officer' plus one of the 100 company names, 57,265 sentences containing 'mayor' plus one of the 100 city names, 904,326 sentences containing the word 'acquire', 'acquisition', or 'acquired', and 927,296 sentences containing the word 'merge', 'merger', or 'merged'.

The underlying IE system for these experiments was KnowItAll with the pattern-learning module turned on. It works by using a set of domain independent extraction patterns (e.g. '<X> is the <REL> of <Y>', '<Y>'s <REL> <X>') to bootstrap a set of seeds, uses those seeds to find relationship specific patterns, and then extracts items based on those patterns. The learned patterns were constrained to have at most 5 tokens between the slots, and at most 3 to the right/left. It additionally merged company names that differ only in a generic corporation identifier (e.g.

'Microsoft' vs. 'Microsoft Corp.'), or mayor/ceo extractions that match both the person's last name and the city/company name. As such, we modified our hybrid systems to merge the corresponding contexts similarly.

In this case, we defined the set of contexts for IE_{tf-idf} and IE_{BM25} as all sequences of up to 20 tokens between the arguments, and any sequence of up to 19 stopwords and one non-stopword to the left or right of the extraction. Other than that, the hybrid system performs exactly as in the NER case above. From the sentences described above, there were 635,334 distinct contexts and an average of 62.4 total contexts per extraction.

We generated Precision-Recall curves for each of the relations, comparing the baseline PMI scoring metric with the IE_{tf-idf} and IE_{BM25} hybrid systems (Figures 4-7). Since there were such a large number of extractions, we estimated the precision by hand-tagging a random sample of extractions at various confidence levels and extrapolated that precision to the entire set. This caused a few visual inconsistencies in the graphs. In reality, all three methods have the same absolute recall (number of correct extractions), but in the graphs, the methods estimate a slightly different number of correct extractions. The reason they are different is due to the differences in the randomly selected samples used to estimate the precision. We attempted to sample sufficiently large sets to provide an accurate comparison, but the small differences caused these minor differences in the estimated recall.

For the relations with one argument bound (MayorOf and CeoOf), the IE_{BM25} hybrid system does a much better job of separating the correct extractions from the incorrect ones than either the baseline or IE_{tf-idf} hybrid system. It especially shines in distinguishing amongst items extracted only one or two times. For the MayorOf relation, the baseline system makes some early mistakes by extracting "Lord" as the mayor from the phrases "Lord Mayor of Dublin", "Lord Mayor of Manchester", etc. Both IE_{BM25} and IE_{tf-idf} have lower confidences in these extractions for two reasons. First, in these cases 'Lord' always appears before 'Mayor', so unlike true mayors, these extractions do not benefit from appearing in any other good contexts. The second reason is that "Lord Mayor" is often preceded by 'the', whereas actual mayor names do not typically occur in the phrase 'the <person name> mayor of <city>'. The IE_{BM25} hybrid system does not have as impressive results with the CeoOf relationship, but since the baseline is so high, it is difficult to improve it significantly. However, it is still provides a better ranking for the infrequent extractions, and so outperforms the baseline at higher levels of recall. Furthermore, it provides a finer distinction amongst extractions, so allows a finer-grained trade-off between precision and recall.

The IE_{tf-idf}, on the other hand, performs worse than the baseline for the CeoOf relation. This is mainly due to the heuristic we used for merging ceo names. We attempted to group mayors and ceos by last name (e.g. so that 'Bill Gates', 'William Gates', 'William H. Gates' are all considered the same). However, for incorrect extractions such as "Peoplesoft CEO Blew Merger Deal", we would use any sentence mentioning 'Peoplesoft' and 'Deal' as evidence for 'Blew Merger Deal' being the name of a ceo of Peoplesoft. The IE_{tf-idf} hybrid system does not normalize for the number of mentions, so this incorrect name matching heuristic generated a large amount of evidence for this incorrect extraction. Yet in spite of this, preliminary experiments showed that simple normalization techniques tended to favor infrequently extracted items, and so tended to generate

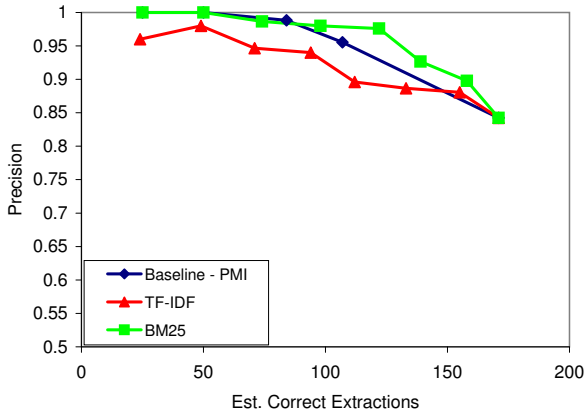

Figure 4: Precision/Recall for CeoOf(CEO, Company)

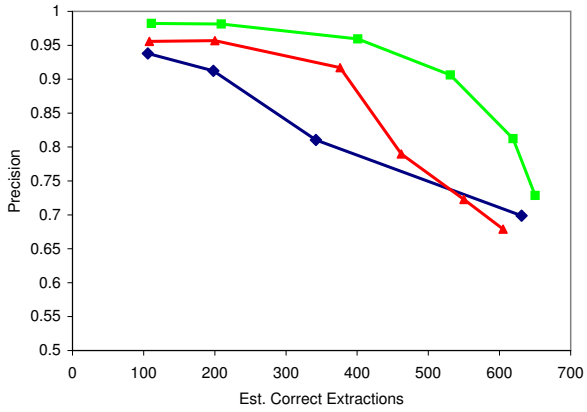

Figure 5: Precision/Recall for MayorOf(Mayor,City)

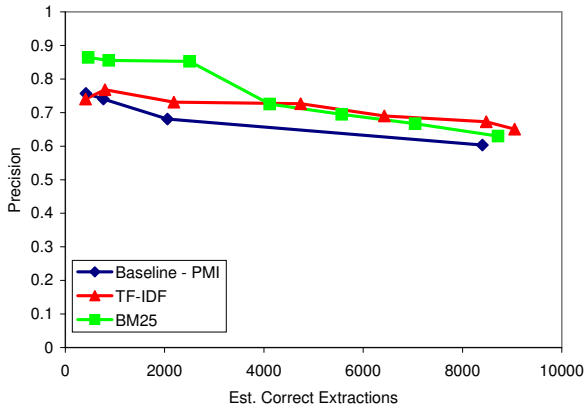

Figure 6: Precision/Recall for Acquired(Company,Company)

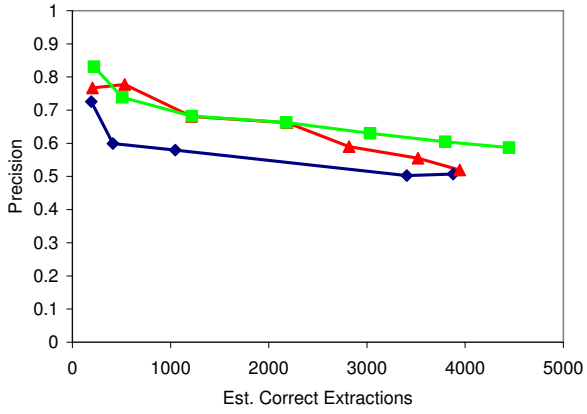

Figure 7: Precision/Recall for MergedWith(Company,Company)

worse results. As the experiments show, IE_{BM25} appears to provide a good method of both normalizing and effectively using frequency data.

For the unbound relations (Acquisition and MergedWith), the baselines were significantly lower. The reasons for this are that a) the data is significantly sparser (about 3/4 of the items are only extracted once or twice), and b) there is much more ambiguity in the extractions, since neither term is known to be of the correct class. We matched corporation names by only ignoring generic corporation identifiers (e.g. corp, inc, company, etc.), and so the IE_{tf-idf} hybrid did not have the problems given above. As a result it performed significantly better.

Since the baseline system only considered 'good' patterns, it made a number of incorrect extractions by ignoring wider contextual evidence. For example, the baseline system is fairly confident that CNOOC acquired Unocal. However, its evidence comes from a large number of sentences like '...likely to block CNOOC's acquisition of Unocal' or 'The proposed acquisition of UNOCAL by CNOOC...' Relatedly, since it uses a simple proper noun recognizer, it is not able to determine that the types of objects are correct. For example, it cannot distinguish that 'Hospital Acquired Infection' is discussing diseases, not corporations. While the IE_{BM25} and IE_{tf-idf} hybrid systems do not have this type information explicitly, they make use of the surrounding context to give those types of extractions less weight.

Just as in the bound argument case and the Named Entity Recognition experiments, these systems tend to better distinguish between correct and incorrect items, especially for items that are rarely extracted.

5 Related Work

There is clearly a large body of related work in Information Retrieval, Information Extraction, and Text Classification. However, since our experiments focus on IE, we will touch briefly on related work in IR and TC, and focus mainly on related work in IE.

Information Retrieval has been studied since the early days of computer science. Since it is such a fundamental and useful problem many techniques have been studied, ranging from simple boolean keyword matching and probabilistic models based on binary word presence/absence information [24], to more complex systems based on language modeling [21], probabilistic networks [5], or hyperlink structure [20]. However, despite their simplicity, techniques combining term and inverse document frequencies have proven to be both efficient and effective. While we chose two particular variants in our experiments, they are but two members of a much larger family of tf-idf term weighting functions.

Similarly, there is a large body of related work in Text Classification. Of particular relevance to this work are [15], which discusses the relationship between Naive Bayes and IR, and [12], which demonstrated that IR weighting and preprocessing techniques can significantly improve the performance of a Naive Bayes classifier. Additionally, [13] uses similarities between text classifiers based on the Naive Bayes model and on a tf-idf document vector distance based model. Through analysis of their similarities, they propose probabilistic tf-idf classifier and show that it leverages the benefits of both.

Our experiments are most closely related to unsupervised techniques in Information Extraction.

The ideas we propose are complimentary to previous work in pattern learning, since we focus on how extraction confidence is computed, and treat pattern definitions and learning as external processes. Thus, the idea of computing confidence based on IR techniques can "plug in" to a wide variety of pattern-based IE techniques and potentially provide similar benefits.

Our results are probably best compliment systems such as KnowItAll [8], Snowball [3], URES [26], and NOMEN [30] which attempt unsupervised extraction from large corpora. While the four systems use different notions of patterns (strings, IR-style term vectors, regular expressions, and string with wild cards respectively), they all compute extraction confidence as some function of how often, how many, and how well the *positive* patterns match. As we argued before, this implicitly discards information about the extractions. Although we only showed the benefits of using this information in the KnowItAllsystem, we believe it should extend to these other systems just as easily.

Unlike those systems, and similar to our experiments, Basilisk [28] scores each extraction based on all patterns it occurs with. Although it uses a particular heuristic which they demonstrate to be effective, whereas we advocate borrowing from IR, the two are similar in spirit. In [10] Hasegawa et. al. attempt to find relationships between pairs of named entities, by clustering based on context. Our experiments, on the other hand, sought to verify the existence of a specified relationship. Recent work by Paşca et. al. [19] extracted over one million facts (name, year of birth pairs) from the web. [19] made heavy use of distributional similarity, using it to determine their patterns, verify extractions, and score them. In their work, they score patterns and extractions based on the best single word matches, whereas our experiments required full phrase matching. Their system, as well as Snowball, URES, NOMEN, and others generalize much better since they do not require strict phrase matching. However, these systems also face the problem of distinguishing between "X's completed merger with Y" and "X's failed merger with Y". For our experiments, we chose to favor specificity over this type of generality, but conceptually the ideas presented here can be adapted to these more general patterns and associated systems.

In [2], Agichtein and Cucerzan use both probabilistic and IR methods to estimate how difficult the general task of extracting names and relations from a particular corpus is, independent of the extracting system. We instead propose using similar techniques to compute a particular extraction's confidence directly.

Finally, supervised IE techniques based on techniques such as Hidden Markov Models [9], Rule Learning [27], Conditional Random Fields [17], or Wrapper Induction [14] explore different efficiency, scalability, and precision trade-offs. By focusing on the simpler notion of pattern-based extraction and leveraging IR scoring functions, we were favoring techniques that can operate efficiently at large scales and with little training data. In contrast, these methods require more domain-specific training data and/or are more computationally expensive. While they have been shown to perform well, they are not the target of this work.

6 Conclusions and Future Work

In this paper we have explored the relationship between Information Retrieval, Information Extraction, and Text Classification. We've argued that, for an interesting subset of each (document-vector

based IR systems, pattern-based IE systems, and linear text classification systems), the main difference is in how they represent the query and the corpus. By using simple transformations between these problems, we were able to explore the inherent assumptions in a tf-idf IR system. Specifically, we showed that a particular tf-idf IR system ranks documents equivalent to a Noisy-Or system, under certain assumptions of the probability of term relevance, and we argued that the IR tf-idf metric behaves similarly to a compliment Naive Bayes classifier.

Finally, we demonstrated some practical benefits of these observations. By ranking extractions based on IR scoring metrics, we have shown that IR functions can be used as a simple, scalable, unsupervised way to improve the precision/recall trade-off of a pattern-based IE system. We demonstrated that, for a variety of Named Entity classes and a variety of relationships between entities, using IR techniques to aggregate all contextual information about an extraction can improve results. These techniques worked by both boosting the confidence of infrequently extracted items, while (relatively) lowering the confidence of incorrectly typed extractions.

There are several interesting avenues for future work. Aside from exploring different IR weighting techniques, we would like to examine if these techniques can benefit other IE systems such as Snowball or NOMEN. Additionally, although our experiments used only a small number of positive examples, some preliminary experiments have demonstrated that using a small set of negative examples as well can further improve performance. Automatically discovering negative examples and determining how to best use them provides intriguing possibilities of future research. Lastly, exploring how ideas from IE can influence IR or TC could have benefits. For example, the recently proposed URNS model [7] in IE uses redundancy to determine, with surprising accuracy, an estimate of the probability that each extraction is correct. Determining methods to appropriately apply it to an IR system might yield interesting results.

7 Acknowledgments

I would like to thank my advisors Oren Etzioni and Pedro Domingos for all of their support, Stephen Soderland for all of his guidance in this work, Douglas Downey for his input along the way, Katarzyna Wilamowska for her comments on this document, and my parents for their constant encouragement.

A Proof of Theorem 3.1

The proof results from substituting the terms into Equation 1

Proof.

$$IR_{tf-idf}(C, q, d_1) > IR_{tf-idf}(C, q, d_2)$$

$$\sum_{j \in V} d_{1j} * q_j * \log \frac{N}{n_j} > \sum_{j \in V} d_{2j} * q_j * \log \frac{N}{n_j}$$

Since $q_i = 0$ for all terms not in the query:

$$\sum_{j \in Q} d_{1j} * q_j * \log \frac{N}{n_j} > \sum_{j \in Q} d_{2j} * q_j * \log \frac{N}{n_j}$$
$$\sum_{j \in Q} d_{1j} * \log \frac{N^{q_j}}{n_j^{q_j}} > \sum_{j \in Q} d_{2j} * \log \frac{N^{q_j}}{n_j^{q_j}}$$
$$\sum_{j \in Q} d_{1j} * \log \frac{1}{\frac{n_j^{q_j}}{N^{q_j}}} > \sum_{j \in Q} d_{2j} * \log \frac{1}{\frac{n_j^{q_j}}{N^{q_j}}}$$
$$\sum_{j \in Q} d_{1j} * \log \frac{1}{1 - \frac{N^{q_j} - n_j^{q_j}}{N^{q_j}}} > \sum_{j \in Q} d_{2j} * \log \frac{1}{1 - \frac{N^{q_j} - n_j^{q_j}}{N^{q_j}}}$$

Substituting the created patterns and probabilities for the query terms:

$$\sum_{p_j \in P} d_{1j} * \log \frac{1}{1 - Pr(p_j)} > \sum_{p_j \in P} d_{2j} * \log \frac{1}{1 - Pr(p_j)} \tag{6}$$
$$\sum_{p_j \in P} d_{1j} * \log(1 - Pr(p_j)) < \sum_{p_j \in P} d_{2j} * \log(1 - Pr(p_j))$$
$$\prod_{p_j \in P} (1 - Pr(p_j))^{d_{1j}} < \prod_{p_j \in P} (1 - Pr(p_j))^{d_{2j}}$$
$$1 - \prod_{p_j \in P} (1 - Pr(p_j))^{d_{1j}} > 1 - \prod_{p_j \in P} (1 - Pr(p_j))^{d_{2j}}$$

By construction, if word w_j appears d_{ij} times with document i, then the sentence 'w_j $d_i.id$.' will appear d_{ij} times in C'. So $d_i.id$ will be extracted d_{ij} times by by the pattern 'w_j <X> .' Thus:

$$IE_{noisy-or}(C', P, d_1.id) > IE_{noisy-or}(C', P, d_2.id)$$

This proves the $\Rightarrow$ direction of Theorem 3.1. Since all the steps are reversible, the $\Leftarrow$ direction follows from simply reversing the steps.

B Proof of Theorem 3.2

The proof is similar to that of Theorem 3.1. It follows from simple substitutions into equations 3 and 1 and some simple algebraic manipulations:

Proof.

$$IE_{noisy-or}(C, P, ex_1) > IE_{noisy-or}(C, P, ex_2)$$
$$1 - \prod_{p_i \in P} (1 - Pr(p_i))^{n_{p_i ex_1}} > 1 - \prod_{p_i \in P} (1 - Pr(p_i))^{n_{p_i ex_2}}$$

Since $n_{p_i ex_j}$ is the number of times pattern p_i was seen with extraction ex_j, and since the construction adds p_i to document d_j once for each of those, we can simplify this notation by using the IR form: d_{ji} for $n_{p_i ex_j}$.

$$\sum_{p_i \in P} d_{1i} \log(1 - Pr(p_i)) < \sum_{p_i \in P} d_{2i} \log(1 - Pr(p_i))$$

Let E be the number of distinct extractions, and e_i be the number of distinct extractions matching pattern p_i. By assumption $e_i > 0$, thus the following is equivalent and well defined:

$$\sum_{p_i \in P} d_{1i} \log(1 - Pr(p_i)) \frac{\log e_i/E}{\log e_i/E} < \sum_{p_i \in P} d_{2i} \log(1 - Pr(p_i)) \frac{\log e_i/E}{\log e_i/E}$$

$$\sum_{p_i \in P} d_{1i} q_i \log \frac{e_i}{E} < \sum_{p_i \in P} d_{2i} q_i \log \frac{e_i}{E}$$

$$\sum_{p_i \in V} d_{1i} * q_i \log \frac{E}{e_i} > \sum_{p_i \in V} d_{2i} * q_i \log \frac{E}{e_i}$$

Finally, we notice that for the constructed corpus C', the total number of documents is equal to the total number of extractions E, and the number of documents with term $p_i \in V$ is just e_i, the number of distinct extractions with that pattern, then this is exactly equivalent to Definition 1.

$$IR_{tf-idf}(C', Q, d_1) > IR_{tf-idf}(C', Q, d_2)$$

This shows the $\Rightarrow$ direction for Theorem 3.2. The $\Leftarrow$ direction is proved since each step and substitution is reversible.

References

[1] E. Agichtein and L. Gravano, *Querying text databases for efficient information extraction*, Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE), 2003.

[2] Eugene Agichtein and Silviu Cucerzan, *Predicting accuracy of extracting information from unstructured text collections*, CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management (New York, NY, USA), ACM Press, 2005, pp. 413–420.

[3] Eugene Agichtein and Luis Gravano, *Snowball: Extracting relations from large plain-text collections*, Proceedings of the Fifth ACM International Conference on Digital Libraries, 2000.

[4] Michael J. Cafarella, Doug Downey, Stephen Soderland, and Oren Etzioni, *Knowitnow: Fast, scalable information extraction from the web*, Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2005.

[5] J. Callan, W. Croft, and S. Harding, *The inquery retrieval system*, Proceedings of the 3rd International Conference on Database and Expert Systems, 1992.

[6] Pedro Domingos and Michael J. Pazzani, *On the optimality of the simple bayesian classifier under zero-one loss*, Machine Learning **29** (1997), no. 2-3, 103–130.

[7] Doug Downey, Oren Etzioni, and Stephen Soderland, *A probabilistic model of redundancy in information extraction*, Proceedings of the Nineteenth International Joint Conference on Artifical Intelligence, 2005.

[8] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates, *Unsupervised named-entity extraction from the web: an experimental study*, Artif. Intell. **165** (2005), no. 1, 91–134.

[9] Dayne Freitag and Andrew Kachites McCallum, *Information extraction with hmms and shrinkage*, Proceedings of the AAAI-99 Workshop on Machine Learning for Informatino Extraction, 1999.

[10] T. Hasegawa, S. Sekine, and R. Grishman, *Discovering relations among named entities from large corpora*, ACL 2004, 2004.

[11] M. Hearst, *Automatic acquisition of hyponyms from large text corpora*, 1992.

[12] Jaime Teevan Jason D. M. Rennie, Lawrence Shih and David R. Karger, *Tackling the poor assumptions of naive bayes text classifiers*, ICML '03, 2003.

[13] Thorsten Joachims, *A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization*, Proceedings of ICML-97, 14th International Conference on Machine Learning (Nashville, US) (Douglas H. Fisher, ed.), Morgan Kaufmann Publishers, San Francisco, US, 1997, pp. 143–151.

[14] Nickolas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos, *Wrapper induction for information extraction*, Intl. Joint Conference on Artificial Intelligence (IJCAI), 1997, pp. 729–737.

[15] David D. Lewis, *Naive (Bayes) at forty: The independence assumption in information retrieval.*, Proceedings of ECML-98, 10th European Conference on Machine Learning (Chemnitz, DE) (Claire Nédellec and Céline Rouveirol, eds.), no. 1398, Springer Verlag, Heidelberg, DE, 1998, pp. 4–15.

[16] A. McCallum and K. Nigam, *A comparison of event models for naive bayes text classification*, 1998.

[17] Andrew McCallum, *Efficiently inducing features of conditional random fields*, Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI03), 2003.

[18] Tom M. Mitchell, *Machine learning*, WCB/McGraw-Hill, 1997.

[19] M. Paşca, D. Lin, J. Bigham, A. Lifchits, and A. Jain, *Names and similarities on the web: Fact extraction in the fast lane*, Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, July 2006, pp. 809–816.

[20] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, *The pagerank citation ranking: Bringing order to the web*, Tech. report, Stanford Digital Library Technologies Project, 1998.

[21] Jay M. Ponte and W. Bruce Croft, *A language modeling approach to information retrieval*, Research and Development in Information Retrieval, 1998, pp. 275–281.

[22] Ellen Riloff, *Automatically generating extraction patterns from untagged text*, AAAI/IAAI, Vol. 2, 1996, pp. 1044–1049.

[23] Ellen Riloff and Rosie Jones, *Learning Dictionaries for Information Extraction by Multi-level Boot-strapping*, Proceedings of the Sixteenth National Conference on Artificial Intelligence, The AAAI Press/MIT Press, 1999, pp. 1044–1049.

[24] S.E. Robertson and K. Spärck Jones, *Relevance weighting of search terms*, Journal of the American Society for Information Science **27** (1976), 129–146.

[25] Stephen E. Robertson, Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau, *Okapi at TREC-3*, Text REtrieval Conference, 1992, pp. 21–30.

[26] B. Rosenfeld and R. Feldman, *Ures: an unsupervised web relation extraction system*, Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, July 2006, pp. 667–674.

[27] S. Soderland, *Learning information extraction rules for semi-structured and free text*, Machine Learning **34** (1999).

[28] M. Thelen and E. Riloff, *A bootstrapping method for learning semantic lexicons using extraction pattern contexts*, Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing, Philadelphia, PA, July 2002, pp. 214–221.

[29] Peter D. Turney, *Mining the Web for synonyms: PMI–IR versus LSA on TOEFL*, Lecture Notes in Computer Science **2167** (2001), 491–??

[30] R. Yangarber, W. Lin, and R. Grishman, *Unsupervised learning of generalized names*, Proceedings of the 19th International Conference on Computational Linguistics (COLING 2002), 2002.