Generalizing from Relevance Feedback using Named Entity Wildcards

Abhimanyu Lad
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
alad@cs.cmu.edu

Yiming Yang
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
yiming@cs.cmu.edu

ABSTRACT

Traditional adaptive filtering systems learn the user's interests in a rather simple way – words from relevant documents are favored in the query model, while words from irrelevant documents are down-weighted. This biases the query model towards specific words seen in the past, causing the system to favor documents containing relevant but redundant information over documents that use previously unseen words to denote new facts about the same news event. This paper proposes news ways of generalizing from relevance feedback by augmenting the traditional bag-of-words query model with named entity wildcards that are anchored in context. The use of wildcards allows generalization beyond specific words, while contextual restrictions limit the wildcard-matching to entities related to the user's query. We test our new approach in a nugget-level adaptive filtering system and evaluate it in terms of both relevance and novelty of the presented information. Our results indicate that higher recall is obtained when lexical terms are generalized using wildcards. However, such wildcards must be anchored to their context to maintain good precision. How the context of a wildcard is represented and matched against a given document also plays a crucial role in the performance of the retrieval system.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Information filtering, Relevance feedback, Retrieval models, Selection process; I.5.2

General Terms

Design, Experimentation, Measurement, Performance.

Keywords

Relevance feedback, adaptive filtering, named entities.

CIKM'07, November 6–8, 2007, Lisboa, Portugal.

1. INTRODUCTION

Adaptive Filtering (AF) is the task of online prediction of the relevance of documents with respect to predefined topics. Based on an initial query or a few example documents, the AF system monitors a stream of documents and selects some of them for the user's attention. The user provides feedback on these documents, which is used by the system to update its query model and present a new set of documents to the user. AF has been studied extensively and large scale evaluations have been conducted as part of TREC and TDT forums [1, 2, 7, 8]. Various algorithms have been applied to incremental learning of user's information needs and logistic regression is considered to be the state of the art due to its learning performance as well as computational efficiency in light of frequent updates to the query model [13].

Irrespective of the learning algorithm used to update the query model, the basic idea is to favor terms from documents that the user marked as relevant and down-weight the terms from irrelevant documents. However, such a scheme biases the query model towards specific documents seen by the user, and hence, towards specific terms that occurred in those documents. Therefore, the system will try to retrieve new documents that contain those same terms, and hence, the same information.

Consider the following snippet – *"...Parker reported that over 120 people have died in various shootings and explosions in Baghdad and Diwaniyah. 36 of them..."*. If the user marks this snippet as relevant, a naive approach would be to add terms like "Parker", "120", "people", "died", "shooting", "explosions", "baghdad", "diwaniyah" and "36" into the query model. However, it is unlikely that the user is interested in the person named "Parker", or specific numbers like "120" and "36". If words like "120" and "36" are annotated as `CARDINAL` (numeric) entities, "shootings" and "explosions" are annotated as `VIOLENCE-EVENT` entities, and "baghdad" and "diwaniyah" are annotated as `LOCATION` entities, then the system has a better chance of inferring that the user is interested in casualty counts (as `CARDINAL` entities), acts of violence (as `VIOLENCE-EVENT` entities), in the above-mentioned or other locations (as `LOCATION` entities).

Consider another snippet – *"...the oil tanker, named Jessica, was 30 years old and..."*. If the user marks this snippet as relevant, and has also marked other snippets in the past that contained the word "Jessica", it is likely that the user is interested in knowing more about this oil tanker. Hence, instead of memorizing words like "30", "years", and "old", the system should try to find passages that contain

other "facts" about Jessica, e.g. its LOCATION and owning ORGANIZATION.

These examples suggest a pattern-based approach to adaptive filtering – instead of memorizing words, induce generic patterns from documents marked as relevant and use these patterns to find other documents that are likely to be of interest. Our patterns, as evident from the above examples, are based on named entity wildcards that favor documents containing entities that the user would be interested in.

2. RELATED WORK

Neither the automatic induction of patterns from text nor the use of named entities to generalize beyond lexical terms are new ideas. However, these ideas have not been combined to improve performance on ad-hoc retrieval and adaptive filtering tasks. Moreover, recent advances in tagging accuracy and the rich set of entities and events that can be identified by modern taggers provide an excellent opportunity to explore their impact on information retrieval tasks like adaptive filtering that have mostly been approached using the traditional "bag-of-words" model.

2.1 Question Answering

Named entity annotations have found significant use in question answering (QA) where a majority of questions (such as *who-*, *where-*, and *when-* questions) have answers in the form of PERSON, LOCATION, DATE and TIME entities [11, 4]. QA systems take advantage of this fact by using surface patterns based on part-of-speech tags or named entities e.g. ``<PERSON> was born on <DATE>'' to answer common questions e.g. ``When was <PERSON> born?''. Such patterns can also be used to extract information from a corpus offline so that frequent question types can be answered efficiently [3].

However, users in AF settings often start with vague queries instead of specific questions that cannot be classified neatly into specific question types. In fact, a user might not even have specific information needs until he or she reads a few articles delivered by the retrieval system. For instance, a user might start with a query like "Galapagos oil spill" to learn more about the oil spill accident. After reading a few relevant documents, the user might get more interested in details about the tanker (how old it was, who owned it) and about the Galapagos islands (their location). Thus, the system must infer the kind of questions that might be of interest to the user, and hence, induce patterns accordingly from the documents marked as relevant by the user.

2.2 Information Extraction

Automatic induction of patterns for information extraction is an important task with applications in many other fields. AutoSlog-TS [5] automatically extracts candidate patterns from untagged documents labeled as relevant or irrelevant to a given domain. The extracted patterns, e.g. attack on <np>, can be seen as wildcards (<np>) with context (attack on) that can capture phrases like ``attack on embassy''.

However, these patterns are more restrictive and will not match variations like ``the embassy was attacked''. This is not an issue if enough training data is available in which case all such variations can be learned. One of the main challenges in AF systems is limited training data. Our wildcards are more flexible in the way they place restrictions on the context. Moreover, we use a richer set of annotations that can differentiate entities like people, locations, organizations, etc., as well as events like violence, demonstrations, disasters, etc., so that fine-grained information needs can be captured effectively.

2.3 Text Categorization

The idea of using generic patterns that match words in context has also been applied to text categorization. Riloff used a training set of documents to induce *relevancy signatures* [6] that consisted of a trigger word and certain linguistic constraints (e.g. presence of "bomb" as active verb). These signatures were matched against test documents to predict their category. Relevancy signatures were further augmented with domain-specific semantic constraints, e.g. the subject must be tagged as TERRORIST ORGANIZATION.

While they use such semantic roles as additional constraints to improve precision, we use named entity tags as wildcards to allow other related terms to match, as a means of improving recall. Moreover, instead of requiring the presence of a single word as a contextual constraint, we represent the context as a vector of terms with associated weights and compute a "soft-match" between the patterns and given documents, so that documents can be ranked by their relevance.

2.4 Thesaurus-based Query Expansion

Thesaurus-based query expansion approaches can also be seen as ways of generalizing the query model. Augmenting the query with synonyms can retrieve documents that contain words semantically related to the terms in the query.

Such an approach does not help when the user is looking for other information related to the event, in the form of other entities like persons, organizations, dates, etc. These "other entities" are not synonymous with entities already retrieved by the system.

3. OUR APPROACH

We wish to explore a pattern-based approach to the adaptive filtering task – given a set of documents that the user marked as relevant, extract patterns from them that will help in finding other documents that the user might be interested in. However, these other documents should contain information that has not already been seen by the user in the past. Hence, instead of memorizing words from documents marked as relevant by the user, the system must infer more generic patterns about the user's interests.

Information that is related to the query but previously unseen by the user can be of two types:

1. Other facts of the *same* type. For example, if the user has expressed interest in the COUNTRY that reacted to an event, the user might also be interested in other COUNTRYs that have reacted to this event. Similarly, user's interest in one VIOLENCE-EVENT in Iraq can be assumed to suggest interest in other mentions of VIOLENCE-EVENT in Iraq.

2. Facts of *other* types. For example, if the user has expressed interest in the AGE of a person, the user is likely to be interested in knowing other facts about this person, e.g. his or her COUNTRY, the ORGANIZATION that this person belongs to, and so on. Similarly,

Table 1: Examples of Wildcards

Type	Wildcard	Description
S-Wildcard	(COUNTRY){Cuba, trade, embargo}	Matches country names close to the terms "Cuba", "trade", "embargo"
S-Wildcard	(VIOLENCE-EVENT){Iraq}	Matches violence-related terms close to "Iraq"
A-Wildcard	(ANY){Jessica}	Matches any terms tagged as entities close to "Jessica"
A-Wildcard	(ANY){Cheney}	Matches any terms tagged as entities close to "Cheney"

user's interest in an oil tanker called "Jessica" can be assumed to suggest interest in other details about Jessica, e.g. `AGE` of the vessel, owning `ORGANIZATION`, etc.

Hence we must use *wildcards* that can match any words that are tagged with appropriate entity types. It is obvious that two different kinds of wildcards are needed to capture the two kinds of information as described above. The first type of wildcards should match entities of a specific type. We call these as *S-Wildcards* (Specific entity type Wildcards). The second type of wildcards can match against all types of entities. We call these as *A-Wildcards* (Any entity type Wildcards).

However, these wildcards would simply match against entities irrespective of their context. For example, a wildcard that matches `AGE` entities will match against any mention of age – be it age of the person of interest, some other person, or even the age of a monument. Hence, we must anchor the wildcards to the context in which they should be triggered.

Table 1 shows some example wildcards of both types. We give details of how these wildcards are induced and how they are anchored to their contexts in section 6. First we describe in section 4 the adaptive filtering framework that we use and how it is crucially different from traditional adaptive filtering setups explored in TREC and TDT forums. In section 5, we give an overview of our various system components.

4. NUGGET-LEVEL ADAPTIVE FILTERING

In real life, the information need of a user is not satisfied by "relevant documents" as such. Instead, it is satisfied by relevant facts, i.e. nuggets of information, or simply nuggets. These nuggets can come from entire documents, passages, or even individual sentences. Moreover, two documents that contain the same nuggets provide no more utility than a single document. In fact, a user wastes time and effort by reading a *redundant* document presented by the system.

Nugget-level adaptive filtering is a framework that evaluates a filtering system at the nugget-level. Thus a system receives credit for presenting spans of text that contain relevant and novel information, and is penalized for presenting redundant or otherwise irrelevant information.

Nugget-level Adaptive Filtering has important implications on the filtering task:

1. Given a nugget-level evaluation, the user is assumed to be looking for specific facts. Such information needs are more likely to be met by relatively shorter passages rather than full news articles that place an undue burden on the user to extract the relevant facts. Hence, the system must present short passages to the user that are most likely to contain useful information.

2. Since redundant information is penalized, the system must determine the novelty of each candidate passage with respect to passages delivered to the user in the past. Passages that are very similar to the ones previously seen by the user should be suppressed by the system.

3. Relevance feedback at the passage level might be too specific to learn from – when the user marks a passage as relevant, he or she is not looking for other passages containing the same information, but instead, wants to see passages containing related information. While novelty detection can help in suppressing redundant passages, the system still needs to be able to generalize from the feedback to determine what other passages the user would be interested in. In the traditional document-level adaptive filtering, this is less of an issue since it is usually enough to retrieve other documents that are topically related to the document selected by the user. However, in nugget-level adaptive filtering, documents that are on topic provide no utility to the user unless they contain new facts about the topic.

5. SYSTEM OVERVIEW

Our system consists of two major components – 1) Adaptive Filtering, and 2) Novelty Detection. The adaptive filtering component is responsible for maintaining a query model, retrieving relevant passages, and updating the model based on user feedback. The novelty detection component determines the novelty of each passage selected by the adaptive filtering component, and suppresses any passage that is too similar to a passage presented to the user in the past.

5.1 Adaptive Filtering Component

The adaptive filtering component maintains a query model for each query or information need of the user. This query model is used to rank new passages, and the most relevant ones are passed to the novelty detection component, which selects a subset of novel passages to be presented to the user. When the user marks some of these passages as relevant, the system updates its query model by adding new words or adjusting the weights of existing words in the model.

We use the Vector Space Model [10], where queries and documents are represented using vectors whose dimensions correspond to individual words weighted using the TF–IDF (Term Frequency–Inverse Document Frequency) scheme [9] so that words that occur multiple times in a query or document, but are relatively uncommon otherwise, receive more importance.

We use the logistic regression classifier which is considered to be the state of the art algorithm for incrementally

learning the query model in adaptive filtering [13]. Each query is considered to be a class; membership to this class corresponds to the relevance of a passage with respect to the query. For training the classifier, the initial user query as well as the positive feedback given by the user are used as positive examples, while negative feedback is used for negative examples. Initially, when only the user query is present as positive example, and no negative feedback is available, a background corpus is used to generate negative examples to train the classifier for the first time. The logistic regression model is regularized with a Gaussian prior to avoid over-fitting on the training data. Further details of using logistic regression for adaptive filtering as well as computational issues are described in [13, 14].

The class model $\vec{w}^*$ learned by Logistic Regression is a vector whose dimensions are individual words and whose elements are the regression coefficients, indicating how influential each term is in the query profile. All passages are ranked using this class model as follows:

$$f_{REL}(\vec{x}) \equiv P(y = 1|\vec{x}, \vec{w}^*) = \frac{1}{(1 + e^{-\vec{w}^* \cdot \vec{x}})} \quad (1)$$

and passages with scores above a threshold (tuned on a training set) are passed on to the novelty detection component.

5.2 Novelty Detection Component

The system maintains a *user history* H of all passages presented to the user in the past. Each candidate passage $\vec{x}$ selected by the adaptive filtering component is compared with past passages to ensure that it is sufficiently different in terms of cosine similarity.

The novelty score of a candidate passage $\vec{x}$ is computed as:

$$f_{ND}(\vec{x}) = 1 - \max_{h \in H} \{\cos(\vec{x}, h)\} \quad (2)$$

The novelty score of each passage is compared to a pre-specified threshold (tuned on a training set) and any passage with a score below this threshold is dropped.

6. GENERALIZING FROM FEEDBACK

In this section, we describe how we extend our adaptive filtering component by augmenting the bag-of-words query models with named entity based wildcards. As mentioned before, we induce two types of wildcards that focus on retrieving slightly different types of information that is related to the user's information need.

6.1 Specific Entity Type Wildcards

The goal of Specific-Entity-type wildcards (S-Wildcards) is to look for other facts of the same type – if the user selected a passage containing a country name, organization name, or mention of a violent act, then look for passages that mention other countries, organizations, or violent acts, respectively.

An S-Wildcard S_i consists of a named entity type e_i and an associated context c_i that determines when the wildcard should match. We use the notation $S_i = (e_i)\{c_i\}$ to denote the i^{th} S-Wildcard.

Given a passage marked as relevant by the user, an S-Wildcard is induced for each type of entities present in the passage. These are added to the query model and matched against future passages to predict whether they contain information that the user would be interested in.

For instance, consider the following snippet from a user-selected passage with named entity annotations shown for ease of understanding:

```
...<PERSON>Parker</PERSON> reported that over
<CARDINAL>120</CARDINAL> people have died in
various <VIOLENCE-EVENT>shootings</VIOLENCE-EVENT>
and <VIOLENCE-EVENT>explosions</VIOLENCE-EVENT>
in <LOCATION>Baghdad</LOCATION> and
<LOCATION>Diwaniyah</LOCATION>.
<CARDINAL>36</CARDINAL> of them...
```

Each entity type is used to create an S-Wildcard, whose context is created using the lexical terms that surround the entity. The following S-Wildcards can be induced from this passage:

- S_1 = (PERSON){reported}
- S_2 = (CARDINAL){reported, people, died}
- S_3 = (VIOLENCE-EVENT){died, explosions, shootings, Baghdad}
- S_4 = (LOCATION){shootings, diwaniyah, baghdad}

For simplicity, here we define context using a three-word window around each entity. Stop-words like "that", "have", "and", "in", etc. are ignored. Also note that only one S-Wildcard is created for each entity type – the contexts of multiple entities with the same type are merged together.

Once we have a set of S-Wildcards for a query, we must be able to match these against future passages to predict how relevant they are to the user. To do so, we extract S-wildcards from the future passages in the same way, and determine how similar they are to the wildcards in the query. If wildcards from the query and passage are highly similar, the passage is likely to contain information that the user would be interested in. The degree of similarity between a wildcard S_i in the query and a wildcard S_j from a passage is computed as:

$$sim(S_i, S_j) = \begin{cases} sim(c_i, c_j), & e_i = e_j \\ 0, & otherwise \end{cases} \quad (3)$$

Thus, the similarity between two wildcards is equal to the similarity between their contexts, given both wildcards are associated with the same entity type.

The similarity $sim(c_i, c_j)$ between contexts can be computed using the following matching rules:

1. **Always-match** – assume contexts always match, effectively, only counting the number of entity-types that are common in the query and the passage. This is the baseline approach which does not anchor wildcards to their context. We include this to determine the role played by context when using entity-based wildcards.
2. **Phrase-match** – represent the context using the exact sequence of surrounding words e.g. "reported-that-over-<CARDINAL>- people-have-died". Thus, two contexts either match completely or do not match at all. However, it is unlikely that future passages will contain phrases that can match exactly with such patterns.

Table 2: Four context matching rules applied to different snippet pairs

Snippet Pairs	Matching rule			
	Always	Phrase	Boolean	Soft
7 people died people in Iraq	0.0	0.0	0.0	0.0
7 people died 34 people died	1.0	1.0	1.0	1.0
7 people died 3 died	1.0	0.0	1.0	0.78
7 people died 4000 troops deployed	1.0	0.0	0.0	0.0
7 people died 35000 people voted	1.0	0.0	1.0	0.25

3. **Boolean-match** – contexts are represented as bags of words and are said to match if they have at least one word in common, i.e. $c_i \cap c_j \neq \phi$. e.g. a `CARDINAL` in a future passage must have at least one of the words "reported", "people", or "died" in its vicinity to match against S_2. However, certain words are less important ("reported" is very common in news articles) and hence should receive lesser credit when matched, as compared to words like "died".

4. **Soft-match** – instead of a binary match, compute the degree of match between the two contexts using cosine similarity $\cos(c_i, c_j)$. This allows the use of weighing schemes (e.g. TF–IDF) to give more importance to content-bearing words in the context. Thus, S_2 will have a greater match against the snippet "45 more people died in...", than "7 people are reported missing...", and no match against "5000 troops have been deployed...".

The total S-Wildcard-based score of a passage $\vec{x}$ with respect to a query q can be computed by adding the similarities between each wildcard in the query and each wildcard in the passage:

$$f_{SW}(\vec{x}) = \sum_{S_i \in q, S_j \in \vec{x}} sim(S_i, S_j) \quad (4)$$

Table 2 shows how different snippet pairs receive different similarity scores when using the above-mentioned matching rules. Note how the first snippet pair receives zero similarity score for all matching rules since the snippets do not contain a common entity type (they contain a number and location, respectively). Also note how soft-matching using cosine similarity favors the overlap of important words like "died" compared to words like "people" which are relatively common and hence less content-bearing.

6.2 Any Entity Type Wildcards

The goal of Any-Entity-type Wildcards (A-Wildcards) is to look for other facts about terms that the user has previously expressed interest in. Since most facts are tagged as entities (e.g. person names, organizations, locations, dates, events), A-Wildcards simply need to find passages containing many entities close to the terms denoting the user's interest, e.g. find passages containing entities close to the word "Cheney", if the user has marked many passages in the past containing the word "Cheney".

Table 3: Terms with highest regression coefficients for the query "Galapagos oil spill"

Term	Regression Coefficient
galapagos	0.763179
tanker	0.587772
ecuador	0.556614
island	0.551921
fuel	0.498486
spill	0.455319
leak	0.327691
gallon	0.325236
ecosystem	0.312904
wildlife	0.259604

An Any-Entity-type Wildcard (A-Wildcard) A_i is represented only by a context c_i, which corresponds to the specific terms that the user is interested in. A-Wildcards do not have an associated entity type, since they match against entities of any type. However, for consistency, we write A-Wildcards as $(ANY)\{c_i\}$.

Unlike S-Wildcards, which are induced from each user-selected passage and added to the query, each query q only contains one A-Wildcard: $A_q = (ANY)\{c_q\}$, where c_q is the set of all lexical terms with a positive regression coefficient in the logistic regression model for query q. Terms with positive regression coefficients are indicators of relevance, while the magnitude of the coefficients denotes the degree of relevance. Thus, by favoring entities close to such terms, A_q finds passages that contain more information about things that the user seems to be interested in.

As the user provides feedback on passages, logistic regression updates the query model so that terms that are most related to the user's query receive large regression coefficients, while unrelated terms receive small or negative coefficients. Table 3 shows ten terms with the highest coefficients in the logistic regression model for the query "Galapagos oil spill". Since terms like "Galapagos" (name of the island) and "tanker" are highly relevant to the query, it is reasonable to present passages to the user that contain more information (in the form of entities) closely related to the island or the tanker. The following passage, for instance, is a good candidate:

```
...an oil tanker carrying <CARDINAL>240,000</CARDINAL>
gallons of fuel products...the galapagos islands
```

```
are <CARDINAL>600</CARDINAL> miles off
the coast of ecuador...
```

where only entities close to "galapagos" and "tanker" are shown for the sake of explanation.

The A-Wildcard A_q is matched against a new passage by computing its similarity with S-Wildcards extracted from the passage as described in the previous section. Since A_q places no restriction on the entity type, similarity between A_q and a S-Wildcard S_j from the passage is simply defined as:

$$sim(A_q, S_j) = sim(c_q, c_j) \quad (5)$$

Again, similarity $sim(c_q, c_j)$ between contexts can be computed using the following matching rules:

1. **Always-match** – assume contexts always match, effectively, only counting the number of entities present in a passage. This is a baseline approach which simply favors passages that contain more entities.

2. **Boolean-match** – contexts are said to match if they have at least one word in common, i.e. $c_q \cap c_j \neq \phi$. Note that this gives equal credit to entities that appear close to highly relevant terms, as well as entities that appear close to only slightly relevance terms.

3. **Soft-match** – sum of the logistic regression weights of terms common in c_q and c_j. Thus, entities that appear close to highly relevant terms receive higher credit.

We do not include Phrase-match since A_q's context vector c_q is a bag of words with no ordering information.

The total A-Wildcard-based score of a passage $\vec{x}$ with respect to a query q can be computed by adding the similarities between A_q and each S-wildcard in the passage:

$$f_{AW}(\vec{x}) = \sum_{S_j \in \vec{x}} sim(A_q, S_j) \quad (6)$$

6.3 Combining scores

For each passage $\vec{x}$, we can compute three scores that determine its relevance to the user's information need:

1. The logistic regression score $f_{REL}(\vec{x})$ (equation 1) that favors passages containing terms topically related to the query.

2. The S-Wildcard score $f_{SW}(\vec{x})$ (equation 4) that favors passages containing other information of the "same type" that the user might be interested in.

3. The A-Wildcard score $f_{AW}(\vec{x})$ (equation 6) that favors passages containing information of "other types" related to terms that the user has expressed interest in.

We compute a single relevance score for a passage using a weighted combination of the above scores:

$$f(\vec{x}) = f_{REL}(\vec{x}) + \alpha \cdot f_{SW}(\vec{x}) + \beta \cdot f_{AW}(\vec{x}) \quad (7)$$

The optimum values of α and β are tuned on the training set of queries.

7. EXPERIMENTS

7.1 Data and Experimental Setup

TDT4 was the benchmark corpus in the TDT2002 and TDT2003 evaluations. The corpus consists of over 90,000 news articles from multiple newswire sources published between October 2000 and January 2001, in the languages of Arabic, English and Mandarin. In this paper, we restrict our evaluation to the 23,000 English documents.

We use the 120 queries and associated answer keys as described in [12] as our evaluation corpus. We divide the 120 queries into a training and test set of 60 queries each. The training set is used to tune the relevance threshold, novelty threshold, and the score combination weights α and β (equation 7) using cross-validation. For our corpus, the optimal values when using soft-match rules for both wildcards were found to be as follows: relevance threshold (0.62), novelty threshold (0.5), α (1.0), and β (0.1). Note the small value for β which compensates for the relatively larger scores (f_{AW}) generated when using soft-match rule for A-Wildcards.

We divide the four-month corpus into chunks of 5 days. The AF system is supposed to return a ranked list of 3-sentence passages at the end of each chunk, receive feedback, and then produce new ranked lists for the next chunk, and so on. We believe this is more realistic than a system that returns documents for the user's attention at arbitrary times and expects feedback before it moves on to the next document. Instead, a user might choose to go through the system's output after every n days, where n can be controlled by the user.

7.2 Evaluation

The system is evaluated on the test set using two metrics – 1) Nugget recall, which simply counts the number of unique nuggets that the system was able to retrieve, and 2) Normalized Discounted Cumulated Utility (NDCU), which estimates the total utility derived by the user after going through each system-returned passage.

Nugget Recall: The answer keys for the queries in the test set consist of 705 nuggets. Nugget Recall simply calculates the fraction of these nuggets that the system was able to retrieve. Since we only count unique nuggets, the system does not receive more credit for presenting passages that contain previously seen nuggets. Thus, Nugget Recall is a good indicator of whether the system was able to generalize from past feedback and retrieve new nuggets that might not share common words with previously seen nuggets.

NDCU: Normalized Discounted Cumulated Utility is a new metric that we proposed in [12], which measures the utility of a retrieval system as the difference between how much the user gained in terms of useful information and how much the user lost in terms of time and energy. The gain derived from a system-returned passage is defined as the number of unseen nuggets contained in the passage, while the loss is defined as a constant cost of reading the passage. NDCU takes into account – 1) number of unseen nuggets contained in each passage, 2) the rank of the passage, and 3) the number of passages returned by the system (to penalize long ranked lists). The reader is referred to [12] for a detailed description of NDCU.

Table 4: System performance obtained using various feedback settings

Feedback Setting	Nugget Recall	NDCU Score
No Feedback	0.54	0.37
Logistic Regression (LR) based feedback	0.60	0.41
LR + S-Wildcards		
using Always-match	0.54	0.37
using Phrase-match	0.61	0.43
using Boolean-match	0.62	0.43
using Soft-match	**0.63**	**0.45**
LR + A-Wildcards		
using Always-match	0.36	0.25
using Boolean-match	0.51	0.36
using Soft-match	**0.65**	**0.46**
Best combination LR + S-Wildcards (Soft-match) + A-Wildcards (Soft-match)	**0.66**	**0.48**

8. RESULTS

In Table 4, we report the performance of the system under the following feedback settings:

- **No feedback** – Only the initial user query is used for retrieval. Passages are ranked by the cosine similarity of their TF–IDF vectors with the TF–IDF vector of the user query.
- **Logistic Regression (LR) based feedback** – Based on passages marked as relevant or irrelevant by the user, the query model is updated using logistic regression, as described in section 5.1.
- **LR + S-Wildcards** – In addition to updating the query model using logistic regression, S-Wildcards are induced from user feedback, which are added to the query and matched against future passages. Multiple ways of matching the contexts are reported.
- **LR + A-Wildcards** – In addition to updating the query model using logistic regression, A-Wildcards are induced from user feedback, which are added to the query and matched against future passages. Multiple ways of matching the contexts are reported.
- **Best combination** – All the three methods of applying feedback are used with the best-performing settings.

8.1 Effect of Wildcards

It is evident that both types of wildcards improve performance over the bag-of-words based feedback mechanism that only uses logistic regression (LR) for updating the query model. These improvements are statistically significant when compared with LR based feedback (p-values of 10^{-3} and $2 * 10^{-5}$ for S-Wildcards soft-match and A-Wildcards soft-match, respectively) under sign test for nugget recall.

The best performance obtained from A-Wildcards is greater than that from S-Wildcards. More analysis is needed to determine whether this is due to the nature of this particular set of queries and the answer keys, or whether our hypothesis about "facts of other types" is stronger than our hypothesis about "other facts of the same type" (cf. Section 3).

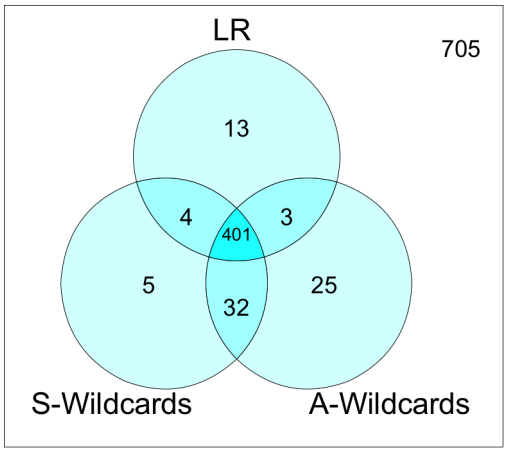

Figure 1: Nuggets retrieved by the three feedback mechanisms when used individually.

To further analyze the contribution of each of the feedback mechanisms, figure 1 shows the Venn diagram of the nuggets retrieved by each of the feedback mechanisms when used alone with the best settings (soft-matching rules). Of the 705 nuggets, there were 401 common nuggets that could be retrieved by any of the feedback techniques when used individually. While 421 (401+4+3+13) nuggets could be retrieved by using LR based feedback, it missed 62 (5+32+25) nuggets that could be retrieved if wildcards were employed. Also, while both wildcards could retrieve a common set of 433 (401+32) nuggets, S-Wildcards could retrieve an additional 9 nuggets which A-Wildcards could not. On the other hand, A-Wildcards could retrieve 28 nuggets which S-Wildcards could not. This reinforces the fact that the contribution of A-Wildcards was larger than that of S-Wildcards. The contributions were not completely overlapping and using both types of wildcards yielded better performance than either of the wildcard types used alone.

8.2 Effect of Context Matching Rules

For both types of wildcards, the best performance was obtained when the soft-matching rule was used for

Table 5: Most frequent entity types in two news events (descending order)

"Gujarat earthquake"	"Texas prison break"
LOCATION	LOCATION
CARDINAL	PERSON
ORGANIZATION	CARDINAL
PERSON	FACILITY
COMMUNICATION-EVENT	ORGANIZATION
DATE	DATE
COUNTRY	COMMUNICATION-EVENT
FACILITY	OCCUPATION
OCCUPATION	VIOLENCE-EVENT
DISASTER-EVENT	WEAPON
VEHICLE	VEHICLE
DURATION	CUSTODY-EVENT

comparing the contexts. Always-matching rule ignores context altogether. However, it does not seem to hurt the performance of S-Wildcards as much as A-Wildcards because in the case of S-Wildcards, the entity types from the passage must still match against those in the query. On the other hand, A-Wildcards place no restrictions on the entity types, causing passages containing more entities of any type to be favored blindly.

9. DISCUSSION AND FUTURE WORK

Two types of wildcards – Although our analysis (figure 1) indicates that both types of wildcards are useful and overall provide the highest performance when used together, we do not expect that both of them are meaningful (or equally meaningful) for every query. Currently, it is not clear how to choose (or favor) one type of wildcard based on the nature of the user's query. Moreover, it might not be essential, after all, to explicitly define the two types of wildcards as we have proposed in this paper – one might be able to create a single abstraction that can model both types of wildcards, and dynamically adjust their relative weights as more feedback is received from the user.

Better use of entity types – In this work, we haven't fully leveraged the rich set of entity and event types that are identified by our tagger. Certain entity types might be more important depending on the nature of the user's query. For instance, table 5 shows the most frequent entity types in two very dissimilar news events. While certain entity types like DISASTER-EVENT and DURATION are more frequent in articles about the Gujarat earthquake, entity types like VIOLENCE-EVENT, VEHICLE and CUSTODY-EVENT are more frequent in articles about the Texas prison break. Based on such observations, one can assign an a-priori distribution on the entity types conditioned on the user's query e.g. giving higher weights to CRIME-EVENT, VIOLENCE-EVENT, and WEAPON entities if the user's query is predicted to be in the "violence/crime" category. User queries can be categorized based on the words in the query itself and the top few documents retrieved by the system for that query.

Also, based on the semantic relationships that hold between certain entity and event types, user's interest in entities of one type could be taken as indication of interest in certain other entity types. For example, user's interest in an ORGANIZATION could trigger a search for CORPORATE-EVENT entities as well; similarly, POLITICAL-EVENT entities might also indicate interest in VIOLENCE-EVENT and DEMONSTRATION-EVENT entities.

Representation of Context – Currently, the context of each wildcard is represented by a collection of words that occurred in the proximity of the entity. We could also allow entity wildcards as part of the context of another wildcard, to capture patterns like "find VIOLENCE-EVENTs mentioned close to any ORGANIZATION entity".

10. CONCLUSIONS

This paper explores new ways of generalizing from relevance feedback through a pattern-based approach to adaptive filtering. Our patterns are essentially named entity wildcards that are anchored to their context. We proposed two types of wildcards that are targeted towards slightly different kinds of information that the user might be interested in. We analyzed different ways in which these wildcards can be anchored to their contexts, and showed that soft-matching rules consistently perform the best. We tested our new approach in a nugget-level adaptive filtering system, and evaluated it in terms of both relevance and novelty of the presented information. Encouraging results were obtained when both kinds of wildcards were used in addition to the traditional bag-of-words feedback mechanism based on a logistic regression classifier.

11. ACKNOWLEDGMENTS

We would like to thank Jonathan Elsas and Sharath Rao for their valuable suggestions and discussions. We also extend our gratitude to the anonymous reviewers for their insightful suggestions on improving the paper. This work is supported in parts by the National Science Foundation (NSF) under grant IIS-0434035, and the Defense Advanced Research Project Agency (DARPA) under contracts NBCHD030010 and W0550432. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

12. REFERENCES

[1] J. Fiscus and G. Duddington. Topic detection and tracking overview. *Topic Detection and Tracking: Event-based Information Organization*, pages 17–31, 1998.

[2] J. Fiscus and B. Wheatley. Overview of the TDT 2004 Evaluation and Results. *TDT Workshop. Dec*, pages 2–3, 2004.

[3] M. Fleischman, E. Hovy, and A. Echihabi. Offline strategies for online question answering: Answering questions before they are asked. *Proceedings of ACL*, 3:1–7, 2003.

[4] J. Prager, E. Brown, A. Coden, and D. Radev. Question answering using predictive annotation. *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR2000)*, pages 184–191, 2000.

[5] E. Riloff. Automatically Generating Extraction Patterns from Untagged Text. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, 2:1044–1049, 1996.

[6] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems (TOIS)*, 12(3):296–333, 1994.

[7] S. Robertson and D. Hull. The TREC-9 filtering track final report. *Proceedings of the 9th Text REtrieval Conference (TREC-9)*, pages 25–40, 2001.

[8] S. Robertson and I. Soboroff. The TREC-10 Filtering Track Final Report. *Proceeding of the Tenth Text REtrieval Conference (TREC-10)*, pages 26–37, 2002.

[9] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. New York, NY, USA, 1986.

[10] G. Salton, A. Wong, and C. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[11] R. Srihari and W. Li. A question answering system supported by information extraction. *Proceedings of the sixth conference on Applied natural language processing*, pages 166–172, 2000.

[12] Y. Yang, A. Lad, N. Lao, A. Harpale, B. Kisiel, and M. Rogati. Utility-based information distillation over temporally sequenced documents. *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 31–38, 2007.

[13] Y. Yang, S. Yoo, J. Zhang, and B. Kisiel. Robustness of adaptive filtering methods in a cross-benchmark evaluation. *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 98–105, 2005.

[14] J. Zhang and Y. Yang. Robustness of regularized linear classification methods in text categorization. *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 190–197, 2003.