Joke Retrieval: Recognizing the Same Joke Told Differently

Lisa Friedland and James Allan
lfriedl@cs.umass.edu, allan@cs.umass.edu
Department of Computer Science, University of Massachusetts Amherst
140 Governors Drive, Amherst, MA 01003-9264

ABSTRACT

In a corpus of jokes, a human might judge two documents to be the "same joke" even if characters, locations, and other details are varied. A given joke could be retold with an entirely different vocabulary while still maintaining its identity. Since most retrieval systems consider documents to be related only when their word content is similar, we propose joke retrieval as a domain where standard language models may fail. Other meaning-centric domains include logic puzzles, proverbs and recipes; in such domains, new techniques may be required to enable us to search effectively. For jokes, a necessary component of any retrieval system will be the ability to identify the "same joke," so we examine this task in both ranking and classification settings. We exploit the structure of jokes to develop two domain-specific alternatives to the "bag of words" document model. In one, only the punch lines, or final sentences, are compared; in the second, certain categories of words (e.g., professions and countries) are tagged and treated as interchangeable. Each technique works well for certain jokes. By combining the methods using machine learning, we create a hybrid that achieves higher performance than any individual approach.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing

General Terms

Algorithms

Keywords

Humor, document similarity, domain-specific retrieval

1. INTRODUCTION

Humor is famously difficult for machines to comprehend. It brings into play ambiguities, implications, and exaggerations, all in the service of violating expectations—which requires one to have expectations in the first place. To believe Hollywood writers, humor will be the last skill for artificial intelligences to acquire. To believe linguistic and computational researchers, jokes are a domain where "the question of semantics can no longer be avoided" [25]; and worse, they contain "language that requires deep conceptual knowledge about the details of human experience" [5].

One concept that humor brings into focus is an alternative notion of document similarity. Even more than is true for the news stories and other informative documents typically used in information retrieval, jokes can be related without having many words in common. What we would consider "one joke" can be retold in vastly different ways.[1] For instance, Figure 1 shows a joke, and Figure 2 outlines how its elements change in other documents in our collection.

Characters can change, the setting can change; it is difficult to describe, at the word level, what it is that stays constant in a joke's structure or meaning. One way to evoke this challenge is to try to formulate a search query for a joke—for instance, to determine if any version of it is present in a given corpus. (Perhaps we hazily recall the joke but want to see a full version before retelling it.) With the example shown in Figure 1, we might begin with "priest rabbi accident wine," but then pause, realizing the joke could really be about any two people, so it would be better to remove them from the query. Next, "accident" could be "crash" or "collision," and "wine" could be "whiskey" or "champagne." What is left? Perhaps a few variations on "drink police accident," a query which is less precise and would still fail to retrieve the version in Figure 3. The problems here are not just synonymy or paraphrasing, but also the difficulties of knowing which aspects of a joke are likely to vary and capturing the wide range of possible alternatives.

Jokes are just one of many domains in which this structural, or logical, document similarity can frustrate searches. Difficulty formulating queries to describe a particular meaning also arises when searching for famous quotations. When one is asked, "Can you find that quote where Einstein said …," sometimes all one can do is verify that Einstein didn't say it, and that in fact no one said exactly that phrase, even though the quote we *wanted* is likely out there. Song lyrics also have this property: one must remember them verbatim to find them on the web. A similar domain is that of proverbs: a given saying may be expressed in numerous ways, particularly across cultures, and it would be interesting to find versions of the same message.

[1] See, for example, the recent documentary *The Aristocrats*, which explores the variations of a single joke [27].

A Rabbi and a Priest are driving one day and, by a freak accident, have a head-on collision with tremendous force. Both cars are totally demolished, but amazingly, neither of the clerics has a scratch on him. After they crawl out of their cars, the rabbi sees the priest's collar and says, "So you're a priest. I'm a rabbi. Just look at our cars. There is nothing left, yet we are here, unhurt. This must be a sign from God!" Pointing to the sky, he continues, "God must have meant that we should meet and share our lives in peace and friendship for the rest of our days on earth." The priest replies, "I agree with you completely. This must surely be a sign from God!" The rabbi is looking at his car and exclaims, "And look at this! Here's another miracle! My car is completely demolished, but this bottle of Mogen David wine did not break. Surely, God wants us to drink this wine and to celebrate our good fortune." The priest nods in agreement. The rabbi hands the bottle to the priest, who drinks half the bottle and hands the bottle back to the rabbi. The rabbi takes the bottle and immediately puts the cap on, then hands it back to the priest. The priest, baffled, asks, "Aren't you having any, Rabbi?" The rabbi replies, "Nah... I think I'll wait for the police."

Figure 1. One version of a joke.

An Irish priest and a Rabbi get into a car accident … The priest asks him, "Are you all right, Rabbi?" The Rabbi responds, "Just a little shaken." The priest pulls a flask of whiskey from his coat and says, "Here, drink some of this it will calm your nerves." … "Well, what are we going to tell the police?" "Well," the priest says, "I don't know what your aft' to be tellin' them. But I'll be tellin' them I wasn't the one drinkin'."

A woman and a man got into a really bad car accident. Both cars are totaled …

There's a guy from ARMY driving from West Point to the Meadowlands, a guy from the NAVY was driving from Annapolis to the Meadowlands, and an Air Force guy who's driving from McGwire in South Jerz to the Meadowlands just to watch the Jets. In the middle of the night with no other cars on the road they hit each other and all cars go flying off in different directions. … The Air Force guy says "Let me see what else survived this wreck." So he pops open his trunk and finds a full unopened bottle of Jack Daniels. …

Figure 2. Variations of the same joke (excerpts).

An English man and an Irish man are driving head on, at night, on a twisty, dark road. Both are driving too fast for the conditions and collide on a sharp bend in the road. To the amazement of both, they are unscathed, though their cars are both destroyed. In celebration of their luck, both agree to put aside their dislike for the other from that moment on. At this point, the Irish man goes to the boot and fetches a 12 year old bottle of Jameson whiskey. He hands the bottle to the English man, whom exclaims, "may the English and the Irish live together forever, in peace, and harmony." The English man then tips the bottle and lashes half of it down. Still flabbergasted over the whole thing, he goes to hand the bottle to the Irish man, whom replies: "no tanks, I'll just wait till the Garda get here!"

Figure 3. Fifth variation, with diverging vocabulary.

A genre closely related to jokes is logic and math puzzles: having solved how to use a five-gallon bucket and a three-gallon bucket to measure out exactly four gallons of water [26] shows one immediately, for example, how to use a 100 mL test tube and a 60 mL test tube to measure exactly 80 mL of hydrochloric acid. In fact, probably every grade school "story problem" fits into a small number of templates; it is easy to generate new problems of a given type (see [28]), but fortunately for teachers, no reverse tool (automatic recognition and solution of homework exercises) is yet known to us. Unfortunately for researchers, neither is there yet a system to refer one to "the same research problem" that may have been studied using different terminology in another field—although a few projects have been aimed at this idea [18].

One final example where structure can matter more than word content is cooking. Websites with recipes can suggest other dishes that contain "chicken" or "green beans." But in the process of learning to cook, it often takes seeing a few examples before we begin to recognize a general technique, e.g., that one can roast pork with peppers using the same steps and the same group of seasonings as for the chicken with green beans. Retrieving other recipes with the same structure would make it easier for novices to learn which aspects one can vary.

In all these domains, we expect search to be difficult because the user cares less about the word content of a document—the information captured by standard retrieval models—and more about the logical relationships among its entities. In this paper, we limit ourselves to studying jokes. Specifically, we focus on recognizing whether two documents are "the same joke," or as we will term it, whether they "match" or belong to the same "joke cluster." This functionality would be a critical component of a joke search engine that incorporates meaning. In response to a query, a results page could list several clusters; for each cluster, it would display one joke and a link to "other versions of this." Creating that list of other versions is the task we address here.

To outline what follows in the paper: Sections 3 and 4 introduce our data and models, including several document models specific to jokes. In Section 5, we consider the task of pairwise classification: given two jokes, decide if they match. Next, in Section 6 we move to a ranking setting, which is closer to our eventual goal: given one joke, retrieve a ranked list of matches. Finally, in Section 6.3 we improve the ranking function by incorporating a classifier. We begin now by situating this project within the context of other research on humor and on domain-specific retrieval; other related work is deferred to Section 8.

2. PREVIOUS WORK

2.1 Humor Studies

There is a small but growing body of research in computational humor. This area typically encompasses two tasks: distinguishing humorous from non-humorous documents, and generating humor. Binsted et al. collects the work of several groups in this field [5]. Some recognition tasks include using text classification to distinguish humorous one-liners from other sentences like headlines or proverbs [14], or recognizing pun-like jokes [19]. One early article with a spirit similar to ours critiques IR techniques as relying too heavily on the specific words in a document; it proposes an architecture for neural networks that would infer the implied context of a sentence and then recognize jokes by the incongruities they contain [25]. Though only a toy system was implemented, the authors do discuss joke retrieval as a domain where the query words may not be found in a document, and where, they argue, one must then include semantics in search. As for humor generation, there are systems that generate pun-based riddles or humorous acronyms [5]; such systems begin with a specific humorous template and automatically instantiate it. Other applications include inserting humor into emails or chatbots as a means to improve human-computer interactions [5], [14]. Finally, there is a search engine for jokes called Jester, but it has been created and studied exclusively as a recommender system [7]. The models of humor that help computers recognize or construct it could be valuable to anyone studying jokes, but none of the above articles considers variations of a single joke, the central idea of this paper.

Outside of computer science, humor research has a long history in fields like linguistics, psychology and philosophy. Mihalcea offers a quick survey of this diverse work [13]; for more information, see the International Society for Humor Studies (http://www.hnu.edu/ishs) and its journal (e.g., [2]). The most influential current theories are the Semantic Script-Based Theory of Humor [16] and its extension, the General Theory of Verbal Humor (GTVH) [2]. Both describe humor as resulting from two contrasting interpretations of the same text. GTVH specifies six properties of a humorous text, from lower-level aspects, like the style and the target (i.e., butt) of a joke, to higher-level aspects, like the contrasting themes and logical mechanism by which the humor unfolds. A recent book by Ritchie offers a strong critique of these and related theories as being poor in testable hypotheses, if rich in intuition [17]. Drawing on methodologies from artificial intelligence and generative linguistics, Ritchie lays out properties a formal model of humor would need to satisfy, and as a first step towards that goal, he begins a bottom-up, descriptive analysis of certain types of texts, among them puns.

In a chapter on joke identity and similarity, Ritchie discusses the "same joke" idea, that certain variations preserve a joke's core identity [17]. This concept was earlier suggested by Hofstadter and Gabor, who described how a given "ur-joke" or "skeleton" can underlie many different jokes [8]. Their example "ur-jokes" permit quite a wide variety of instantiations, as long as the key joke entities and their logical relationships are present. Ritchie points out that one can view jokes either as consisting of a central core with variations, or as having degrees of similarity to a great range of jokes, across several axes of variation. The latter view fits in with GTVH, in which the six properties are said to form a strict order. Changing the joke's target, for example, would purportedly create a variation farther from the original than if one changed the joke's narrative style. In the present work, we use an ad hoc rule to describe the jokes we consider to be the same, but the situation is not clear-cut (see Section 3).

This project is unusual for combining humor studies with information retrieval. In the course of building good statistical models for jokes—in particular, models to identify sets of jokes having the same meaning—we may expect to find new insights about jokes themselves. In addition, for a computational area whose practical utility has been doubted [17], a jokes search engine may be a good motivating application. A search engine that could retrieve joke clusters in response to queries containing possibly different words would enable users to effectively search through jokes, something not currently possible. Such a system would demand joke recognition for its spidering step, as well as the identification of matching jokes, our current goal, for organizing the results of a query.

2.2 Domain-Specific Retrieval

A search engine for jokes is an example of a domain-specific retrieval system. Previous authors have discussed the advantages of specialized systems for performing complex, domain-specific queries on structured data gathered from the web—for instance, on collections of research papers, movie show times, or airplane flights [9], [12]. Certainly the domains mentioned in the introduction (quotations, puzzles, etc.) could all be easier to navigate if they had specialized search tools. In the articles above, the main challenges of building such systems consisted of efficiently spidering the web and recognizing informative documents (a task we sidestep by using existing collections), and of correctly extracting the important fields from the text. Our situation is different: we do not know what information to extract. As noted above, there is no working model of what matters for a joke's identity; there is not even a good intuitive model. So, we shall begin by using language modeling, which after all, performs well for that very complicated text domain of natural language. We also build models that capture structures we expect to be important to a joke's identity: namely, the punch line, and the (abstractions of) entities that appear in the joke.

The approaches and task definitions we use here may inform work in other domains where the word content can vary widely without affecting the meaning. In particular, the notion of abstracting the entities, along with any future techniques for incorporating semantics (to the extent this turns out to be necessary), will be applicable to such domains. In Section 8 we discuss other technical approaches that could potentially be brought to bear on joke retrieval and these related tasks.

3. CORPUS

The corpus consists of approximately 11,000 jokes. These were downloaded from 13 joke archive sites on the web. It was important for the corpus to contain multiple versions of a number of jokes; to increase the odds of such repetitions, several specialized collections were included, such as music jokes and profession jokes, that seemed likely to include repeats.

A large number of the documents contained humor outside the scope of the jokes we wanted to study. We manually removed items like one-liners (which included "yo mama" jokes), quotes, funny but true stories, sarcastic commentaries, "top ten ways to …," and lists. The remainder consists of things like narrative stories (like in Figure 1), light bulb jokes, and question/answer jokes (e.g., "Q: What do you call 5000 dead lawyers at the bottom of the ocean? A: A good start!" or "Q: What do you call a snail on a ship? A: A snailor!"). Duplicate and near-duplicate documents (e.g., those that became identical after stemming and stopping) were also removed.

Sixty clusters of jokes were labeled manually. This was done by creatively constructing queries to find matches for particular jokes. (For humans, this was not difficult, but recall was imperfect: in several cases, the retrieval systems found matches that the authors had missed.) Most jokes do not appear to have matches, but the corpus certainly contains more clusters than those we labeled. The clusters range in size from 2 to 13 jokes, and they include a total of 217 documents.

Judging whether two jokes match can be subjective. As a rule of thumb, we labeled them as matching if one might easily say, "I know *that joke*, except in my version [something varies]." However, there are many ambiguities. For instance, consider light bulb jokes. They might be characterized as a single cluster, if only there were not thousands of them: "How many [people of some type] does it take to change light bulb? [More than one], because [they have some particular property]." At the same time, a rewrite into a non-light bulb joke poking fun at the same property—a transformation that would otherwise seem minor—might be seen as changing the joke, since the light bulb genre is such a recognizable, fixed form. For these reasons, we avoided labeling light bulb jokes and other "difficult" jokes altogether.

In the corpus as a whole, almost half the jokes are just two sentences long. Those jokes we labeled tended to be longer stories, averaging about 12 sentences. This was probably a bias in labeling, and it could imply that the results on the short jokes will be those most representative of future performance. However, it is also possible that the same bias—perhaps, that longer jokes were more interesting to look for, and that shorter jokes, often word puns like the "snailor," were harder to vary—would affect the queries of future users.

4. METHODS

We use a language modeling approach. The document models and similarity measures described next are employed in both the classification and ranking tasks. As noted earlier, we use a standard statistical language model as a baseline [11]. Then, we implement variations that specially treat those structures we expect to be important to a joke's identity.

4.1 Document Models

4.1.1 Baseline

The baseline is a standard unigram (bag of words) model. With this, each document is initially represented as a multinomial probability distribution over its words. The probabilities are estimated using maximum likelihood. That is, if word w occurs $tf_{w,d}$ times in a document d having length L_d, then in the document model M_d,

$$P(w \mid M_d)_{MLE} = \frac{tf_{w,d}}{L_d} .$$

To avoid assigning any words probability zero, we use linear interpolation smoothing to combine the above value with the probability of the word in the general corpus:

$$P(w \mid M_d) = \lambda P(w \mid M_d)_{MLE} + (1-\lambda) P(w \mid M_c)_{MLE} .$$

We determine λ through a parameter sweep, performed separately for each model and task. In the ranking setting, we find the value $\lambda = 0.4$ to be optimal for all models; for classification, we find $\lambda = 0.99$ to be near-optimal for all models.

Throughout this paper, the query is also a document from our collection. However, we do not need to smooth the query model, so we just use the maximum likelihood value for a query q:

$$P(w \mid M_q) = P(w \mid M_q)_{MLE} = \frac{tf_{w,q}}{L_q} .$$

4.1.2 Punch Line

The first alternative to the baseline captures the intuition that the ending of a joke is crucial to its identity and is likely to remain constant despite the rest changing. For this punch line model, we simply identify the last sentence and throw away everything before it. In shortening the document, we are losing information; however, we speculate that the final sentence contains the "key concepts" for the joke, which will help target the search [3]. The same equations above are used, but every document in the corpus is truncated.

4.1.3 Annotations

This approach is motivated by the idea that if the characters, setting and other details can change in a joke, then perhaps we could recognize those changeable elements and replace them with abstractions. For instance, at an abstract level, the joke from the introduction might read like this: "A person and a person were traveling in vehicles that collided." We create such a representation by recognizing certain words and "annotating" them with their category. Using this representation, our judgments of joke similarity might improve for two reasons. First, the annotated words will now match: among jokes in the same cluster, these words may correctly match where the original text did not. Second, the un-annotated words will be informative when examined separately from the annotated words. This set will include both generic words and unusual words; we hope it will be distinctive within each joke cluster.

The following text shows a joke from our corpus and its annotated version (after stopping and stemming):

"Q: What's the difference between a dead snake in the road and a dead lawyer in the road?
A: There are skid marks in front of the snake."

"differ dead #animal[snake] #location[road] dead #person[lawyer] #location[road] skid mark front #animal[snake]"

As one might imagine, when using these annotations (and ignoring the words inside the brackets), the above joke matches identically to another that begins: "Q: What's the difference between a dead dog in the road and a dead politician …"

To implement the annotations, there are two aspects to decide: (a) how to annotate the text, and (b) how to treat the annotated text. For the first question, we create word lists for ten categories (see Table 1) using the web as well as gazetteers included with the information extraction tool GATE (http://gate.ac.uk). During preprocessing, any document word that matches a list word is tagged (respecting some order of precedence for the lists). This is a coarse method and yields obvious markup errors, for example with homographs and irregular plurals, but such problems are present already in the bag of words model. It would also have been possible to create the word lists using WordNet. Such an approach would be easier to generalize to other domains and other categories. But the manually constructed lists are sufficient for a first pass; in addition, they are easy to modify, which lets us correct the more salient markup errors.

Table 1. Categories of annotations.

animal	number
color	organization
currency	person
location	time/date
music	vehicle

Once the documents are annotated, there are a number of options for how to treat the new tokens. A model could be used that treats "#animal[dog]" as similar but not identical to "#animal[snake]." This would be similar to a translation model, as we will discuss in Section 8. Instead, we choose to treat all "#animal[]" tokens as identical. A translation model giving different probabilities for each substitution would behave midway between treating the tokens as distinct, as in the baseline, and treating them as identical, so we place the annotations model at that second extreme.

Formally, for a plain, un-annotated word under the annotations model, $P(w|M_d)_{MLE}$ is as before. For a word w annotated from word list A, the probability becomes

$$P(w \mid M_d)_{MLE} = \sum_{a \in A} \frac{tf_{a,d}}{L_d}.$$

4.1.4 *Combination Models*

Once the documents have been annotated and subdivided into punch line and non-punch line portions, it is easy to invent additional document models that use this same information differently. For instance, one can use only the punch line, but use the annotations model within it. Or rather than using the annotated tokens within the bag of words, one could simply delete them, in the spirit of treating them like stop words; after all, almost every joke probably contains a "#person." In the realm of possible but probably unhelpful models, one can treat a document as a bag of just two types of tokens: punch line and non-punch line words; or, annotated and non-annotated words. Or, to test the conjecture that only some annotation categories are useful, one can choose to use some types of labels but not others, for instance treating all "#animal" tokens as identical, but ignoring "#location" tags and reverting these to the original words.

In our code base, we provide a flexible syntax for specifying document models along the above lines, and we create 108 such variations. The scores from these models are given as inputs to the machine learning classifier introduced in Section 5.3.

4.2 Similarity/Ranking Measures

To measure the similarity of a query to a document, we use the Kullback-Leibler (KL) divergence of the query and document models. This measure is used to rank the documents during retrieval (Section 6) as well as to evaluate the similarity of two documents (Section 5). KL divergence is a natural (though asymmetric) measure of the distance between two probability distributions; it is zero when the distributions are equal and positive otherwise. When the query is held constant, as in the retrieval setting, KL divergence is rank-equivalent to cross entropy, $H(p,q)$, as shown here [10]:

$$\begin{aligned} KL(M_d \parallel M_q) &= \sum_{w \in q} P(w \mid M_q) \log \frac{P(w \mid M_q)}{P(w \mid M_d)} \\ &= \sum_{w \in q} P(w \mid M_q) \log P(w \mid M_q) \\ &\quad - \sum_{w \in q} P(w \mid M_q) \log P(w \mid M_d) \\ &= -H(q) + H(p,q) \\ &\overset{rank}{=} H(p,q) \end{aligned}$$

Often, the summation in the formula is taken over all words in the vocabulary. Since our query model is not smoothed, $P(w|M_q)$ (and thus the whole term) is zero for words outside the query.

The function above allows different weights (probabilities) for the query terms, as well as for the document terms. It is necessary to use a function with this property since in our framework the query is always a full document, not just a few distinct words. When the query weights are all equal, cross entropy reduces to standard query likelihood.

5. CLASSIFICATION

In the classification task, the system is given two documents, and it must determine whether they are variations of the same joke. We set this up as for a machine learning task—creating separate training and test sets and using cross validation—even though most models only "learn" a cutoff threshold. The training and test sets contain positive and negative examples, the positives being joke pairs that match, and the negatives being joke pairs that do not match.

5.1 Training and Test Sets

The samples are divided into ten groups to allow ten-fold cross validation. In order that the training and testing barrier be kept intact, no joke cluster contributes examples to more than one group. We also avoid letting any one large cluster dominate the examples, sampling no more than 15 positives and 15 negatives from any cluster.

For any cluster, the positive examples are drawn from all pairs of jokes in the cluster. The negative examples have one joke in the cluster and one outside it. If the joke from outside the cluster were picked uniformly at random, the task would be unfairly easy; the pair of jokes would not be at all similar. So instead, we sample negatives so that they will be comparable in their *ranks* to the positives. That is, for each positive pair, we use one joke as a query, retrieve a ranked list of jokes, and record the rank (in that list) of the second joke. By repeating this with every joke as the query, we estimate a distribution of ranks of positives. Then, to generate negatives, we take one joke from the cluster, retrieve a ranked list of jokes, sample a desired rank from our distribution, and pick a *non-matching* joke from at or near that rank. In this way we create negative examples that are, in theory, difficult to distinguish from the positives.

5.2 Symmetric Similarity

We described KL divergence above. However, when the example at hand is a pair of documents a and b, with neither taking the role of query, it is better to measure their similarity using a symmetric

score. We make the score symmetric by taking the average of both directions, that is, using:
similarity = $\frac{1}{2}(KL(M_a \,||\, M_b) + KL(M_b \,||\, M_a))$.

It would have been possible to use the symmetric cross entropy instead. Since the values of the scores matter, not just the rankings, we choose KL divergence because it has a minimum of zero. For cross entropy, the minimum score (occurring for perfectly matching documents) is the entropy of the query, which varies by query.

5.3 Experiments

In total, we have approximately 600 data points, of which 58% are negatives. During the training phase, the classifier computes the similarity score for each pair, then it chooses a decision threshold to maximize its accuracy—the number of correct predictions—on the training data. We evaluate the accuracy for each fold of the test data and then compute an average across the folds. Table 2 shows the accuracies achieved by the three main document models described above.

Table 2. Classification accuracy of individual models.

Document model	**Accuracy**
Baseline	0.749
Annotations	0.773
Punch line	0.801

The first things to observe are that the accuracies are fairly high, and that the models that use joke structures have some advantage over the baseline. Also, there is diversity among the models; Table 3 shows how each model has some examples that only it predicts correctly. We further see that the models are erring on the side of caution by not recognizing positives when they appear.

Table 3. Diversity among classification models.

Document model	**Number of pairs only this model gets right**	**Accuracy on negatives**	**Accuracy on positives**
Baseline	4	0.91	0.52
Annotations	13	0.91	0.59
Punch line	56	0.90	0.66

To take advantage of the diversity among the models, we try combining them using machine learning. We use the similarity scores from the models as inputs to a classifier and allow the classifier to make the prediction. We use Weka's logistic regression tool [20]; its other classifiers performed similarly or worse. We test several combinations of features, beginning with the scores from the three models we have seen above. Next, hypothesizing that relative document lengths may be predictive, we add two more features: the ratio and average of the document lengths. Finally, we use as our features the scores from all 108 model variations described in Section 4.1.4.

The results of the classifiers are shown in Table 4. We see that the classifier that uses the set of three features (top line) achieves better performance than any individual model. Adding additional features does not help; if anything, it was useful to manually select the set of three features.

Table 4. Classification accuracy of combination models.

Features	**Number of features**	**Accuracy**
Baseline, annotations, punch line	3	0.818
Above, plus ratio and average of document lengths	5	0.802
Various	108	0.801

We assess significance using paired t-tests on the sets of individual predictions. At the p = 0.02 level, annotations beats baseline, and the best classifier beats annotations; however, for the punch line versus annotations and for the classifier versus punch line, they just miss significance, yielding p-values around 0.06.

It is surprising that the punch line model performs so well here; in light of the poor scores we will see for it in Section 6.2, it is also somewhat misleading. Further analysis suggests that this model's high accuracy in classification is an artifact of the sampling procedure: by intent, we chose negative examples whose scores under the baseline model closely matched the scores of the positive examples. As a result, the baseline model has difficulty distinguishing the classes. The annotations model has a similar property. However, the punch line model tends to give different scores than the other two; thus its positive and negative examples were not pushed together by the choice of samples, and it could outperform the other models in this setting.

6. RANKING

We next consider this "same jokes" task in a ranking setting. Ranking is a more appropriate setting for evaluating the task if we anticipate using the system to retrieve "more jokes like this."

6.1 Setup

In this setting, we use one joke as a query, and we use one of the document models described earlier to rank all the documents in the collection. The relevant documents for this query are defined as those jokes in the same cluster. We measure average precision, recall at various cutoffs, and R-precision. We repeat this process for every joke in the cluster, and calculate the average of the measures for the cluster. After doing this for every cluster, finally we report the averages across all 60 clusters.

6.2 Results

The results of the ranking experiments are displayed in Table 5. We see that the order of performance is reversed from the classification setting; here, the baseline model performs best and the punch line model worst. This holds across all four measures. The differences between the baseline and annotations models are, however, not significant.

Table 5. Ranking performance of individual models.

Document model	MAP	R-precision	Recall at 10	Recall at 100
Baseline	0.793	0.744	0.860	0.966
Annotations	0.774	0.713	0.847	0.948
Punch line	0.514	0.458	0.587	0.737

One way to compare the performance of the models is with a scatterplot of their scores, as in Figure 4. The plots show how closely the annotations and baseline models track each other, as their scores lie near the diagonal (Pearson correlation = 0.84). They also show how the baseline model almost always gives better results than the punch line model. However, we can also see that for each alternative model, there are some clusters in which it soundly beats the baseline. This diversity suggests that again there is potential for improvement by combining the scores of the three models.

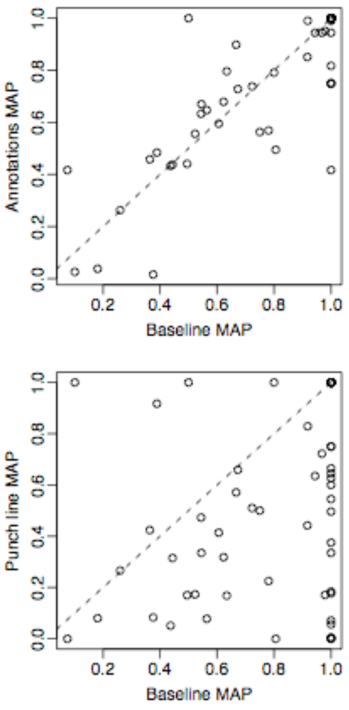

Figure 4. Mean average precision of each joke cluster (one data point per cluster). Diagonal shown for reference. Top, baseline model versus annotations. Bottom, baseline versus punch line.

6.3 Re-ranking

To combine the models, we return to the approach from above: training a pairwise classifier using scores from the three models. The Weka classifier outputs a probability score, not just a binary decision, so we can use this score for ranking. In order to use a pairwise classifier in the ranking setting, where the query is fixed, we have two immediate possibilities. First, we could pair the query with every other document in the collection, one by one, and use the classifier's scores to rank all the documents. Or, we could take some set of top documents from the baseline model and use the classifier to re-rank them. We take the latter approach, for efficiency reasons, and also to exploit the fact that the baseline classifier already has high recall.

To choose the number of documents to re-rank, we plot in Figure 5 the recall curve as a function of the number of documents. The curve levels off by 500 documents, at recall = 0.998.

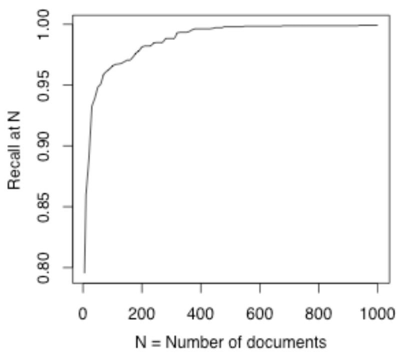

Figure 5. Recall of the baseline model, averaged over all jokes.

In order to train a classifier to re-rank the top 500 documents, we must create a new training set reflecting the distribution where the model will now be applied. For the positives, we use all 442 pairs of jokes in all clusters, since we need all the positive examples we can get. To generate the negatives, we run the baseline ranking, identify the top 500 documents, and sample randomly from them. We use a ratio of about 1:2 for positives to negatives, which keeps the size of the training set reasonably small. (We do not expect it to be important to keep constant the ratio of positives to negatives from training to test sets since we are using the model's output for ranking, as opposed to for classification.)

To create training and test splits, we divide the data into 10 groups of clusters for cross validation. For each cluster, the training data are the positives and negatives from the queries in the other 9 groups.

Table 6 shows the results of using the classifier to re-rank the top 500 documents. (The baseline model, when restricted to its top 500 hits, gives the same scores as in Table 5.) This classifier, when used by itself, performs worse than the baseline. Once more, we examine the scatterplot of scores (Figure 6, top). This time we see that while the classifier does not perform as well as the baseline overall, it is a toss-up as to which works better for any particular cluster. This means that yet once again, we stand to benefit by combining these methods.

Since the machine learning classifier has already been given the baseline score as a feature, we create this final combination by simply linearly interpolating between the output score of the classifier and the baseline score, giving them equal weight. This resulting ranking turns out to be significantly better than any of the others. The bottom of Figure 6 shows how, with the interpolated classifier, the mean average precision of almost every joke cluster improves compared to the baseline.

Table 6. Ranking performance using classifier to re-rank.

Document model	MAP	R-precision	Recall at 10	Recall at 100
Baseline top 500 re-ranked with classifier	0.749	0.684	0.841	0.965
Baseline top 500 re-ranked with (0.5 classifier + 0.5 baseline)	0.822	0.772	0.882	0.977

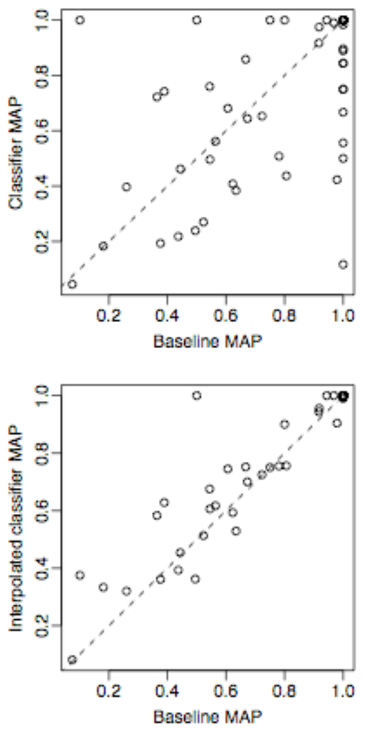

Figure 6. Mean average precision of each joke cluster (one data point per cluster). Top, baseline model versus classifier. Bottom, baseline versus interpolated classifier.

We performed a few experiments analyzing the contribution of the classifier, and in particular, testing whether the improvement in score could be achieved in some simpler way. The results of these experiments are shown in Table 7. One method for improving retrieval in many situations is to expand the query using pseudo-relevance feedback. We created such an expanded query using linear interpolation between the original query and the top *t* documents [23]. We used *t* = 2, and weighted the original query and the new terms 0.4 and 0.6, respectively. Its performance is virtually identical to the baseline.

Next, we investigated whether the boost from the classifier could be due to it using the symmetric version of KL divergence. For this run, we use the baseline model but use the symmetric version of the score. This by itself is clearly not helpful either.

Table 7. Other experiments.

Document model	MAP	R-precision	Recall at 10	Recall at 100
Baseline with pseudo-relevance feedback	0.795	0.740	0.851	0.974
Baseline using symmetric score	0.594	0.534	0.711	0.841

7. ANALYSIS

From these experiments we have learned that the annotations model performs fairly closely to the baseline bag of words model, while the punch line carries differing information. In the classification setting, the task is difficult for the baseline by design, so the punch line model scores well through its contrast. In the ranking task, where the comparison is more fair, the baseline prevails over the other two models. In both settings, we achieve the best results by combining the three document models.

We gain some insight into the utility of the three models by looking at specific queries where they performed differently. Our intuition was that since words from the query would not necessarily appear in the relevant documents, the baseline model would have low recall. For the most part, it seems that if a joke is sufficiently long, certain words actually do appear in all its versions. In the challenging-looking joke cluster from Figures 1–3, for example, the baseline model gives a reasonable MAP for ranking of 0.62; the annotations model scores mildly higher. When a joke is short, the baseline model may still perform well provided there are distinctive words that appear in every version. For instance, the unusual words "trampoline" and "tire gauge" in the joke versions in Figure 7 allow the baseline model to retrieve these clusters perfectly.

What's the difference between a viola and a trampoline? You take your shoes off to jump on a trampoline.

Q: What's the difference between a viola and a trampoline?
A: You don't have to take your shoes off before you jump on a viola.

What's the difference between a bassoon and a trampoline? You take off your shoes when you jump on a trampoline.

Q: How does a blonde measure his/her IQ?
A: With a tire gauge! (da da dum)

Q: How do you measure a blonde's intelligence?
A: Stick a tire pressure gauge in her ear!

Figure 7. Joke clusters easy for the baseline.

There is a mild indication that joke length correlates with the success of the annotations model. In particular, for the cases where the annotations model works better than the baseline, the joke is either short (under 50 words) or long (over 120). For jokes of medium length, either the two models give comparable scores, or the baseline model wins. We can explain the success of the annotations model at short jokes by referring back to the example from Section 4.1.3 involving "skid marks;" in cases such as this, there are not always enough words preserved for the baseline model to use. For instance, in the example in Figure 8, the annotations model scored perfectly. The baseline model had a MAP of 0.5; it found the correct documents by rank 2, but retrieved other tiger and polar bear jokes (respectively) as its top matches.

Q: What's black and white and bounces?
A: A polar bear on a pogo stick!

Q: What's striped and bouncy?
A: A tiger on a pogo stick!

Figure 8. Joke cluster easy for annotations, difficult for the baseline.

As for punch lines, when the punch lines match closely, this seems to be a sufficient condition for the jokes to match. However, this only happens for some jokes.

Overall, it seems as though every joke has some invariant phrases. However, it is difficult to describe, without actually looking at the joke, which phrases those might be. This is why using a combination of methods makes sense: each deals well with certain types of jokes.

8. DISCUSSION

In terms of other possible methods for recognizing joke variants, we considered viewing variants as if they were translations into other languages and then learning a translation model of common word substitutions [6]. This is similar to Berger and Lafferty's use of translation models between (English-language) queries and documents, designed to help connect words having the same meaning or topic [4]. However, those models require a large amount of training data (matched documents), whereas our set of labeled documents, on the contrary, is quite small.

The idea that most joke clusters have particular invariant words or phrases relates to the idea of "key" or "core" concepts, introduced by Allan et al. [1] and recently further developed for use with verbose queries [3]. Even absent any intuitions about jokes, this "key concept" idea would seem relevant because our queries are long—entire jokes; Bendersky and Croft argue that extraneous concepts tend to hinder retrieval performance [3]. It is not clear that concepts which are key in standard text—e.g., proper nouns—would play the same role in jokes, nor that we would have enough data to learn to identify the important terms. However, it would be interesting to try modifying these techniques for jokes.

One possible approach for handling queries whose terms may not appear in the relevant documents comes from work on "vague queries." This was introduced by Motro for the database community [15], but it could be seen as a type of query expansion. The idea is that if a query returns no matches, it can be broadened by examining the closest matches in the corpus. For structured databases, refining the query requires having an appropriate similarity measure for each type of field—for instance, geographic proximity for cities but temporal distance for times. Zhu et al. pose an analogous problem in information retrieval [24]. They describe the challenge of searching for "that book about the guitar-playing sergeant," when the desired title is actually "(Captain) Corelli's Mandolin." Since the data type in this case is words, the work uses a similarity measure defined over WordNet to suggest candidate modifications of the query terms. Among the many possible expansions or substitutions for the query, modified queries are judged good (as opposed to vague) if their terms frequently appear close together in the corpus at large. In an earlier initiative, Woods et al. address this same situation, dubbing it the "paraphrase problem" [21], [22]. Their approach involves building a "conceptual index," a large semantic taxonomy describing relationships among words and phrases. Given a query, the system searches among candidate modifications and generalizations of the query terms. The quality of a new query is judged both by the proximity of the terms within the retrieved documents and also by the similarity of the new query to the original.

These approaches could be promising for querying for jokes, but we see a few drawbacks. First, they would be useful for retrieving *some* matches to a joke, but since they choose combinations of new query terms that are popular in the corpus, their recall could be low. Second, even for short queries, there might be an intractably large search space of plausible substitutions in the jokes domain. To state this more plainly, jokes are not just paraphrases. Paraphrasing might in fact describe our difficulties in searching for song lyrics and quotations. But for jokes and puzzles, it will not be enough to consider synonyms and related terms; entities can shift broadly in different versions, and large swaths of details can be modified or dropped.

It is an open question whether, and to what extent, semantic processing needs to be added to statistical models of language [22]. For identifying "the same" joke, intuition suggests that we would need, at a minimum, information extraction for all the entities, events, and logical relations (each possibly implicit) in a joke—capabilities far beyond today's reach. Yet in many cases in our corpus, it seems to be raw words that matter, essential phrases like "skid marks." Perhaps such words are informative because the corpus is of limited size, because distinctive phrases tend to be preserved in transmission, or because these phrases in themselves define the identity of the joke. Regardless, jokes are yet another domain where the bag of words model performs surprisingly well. Even when they are combined with our other models, however, there is much room left for improvement.

We have used knowledge of a particular domain to build a retrieval system that performs better at ranking and classification than the standard model does in this domain. Along the way, we have used the domain, humor, to argue for alternative definitions of similarity between documents: that they exist and that they matter. In particular, documents in some domains are difficult to search for at present because one cannot be certain of any words the item will contain; only their relationships count.

For a person learning a foreign language, the standard advice goes that they will have mastered it only when they can tell jokes in the language. For computers processing human language, perhaps humor will serve as that same challenge and yardstick.

9. ACKNOWLEDGMENTS

Our thanks to Mario Di Marzo for collaborating on an early version of this project. David Jensen provided support for this work, and also suggested the "same research problem" analogy. Thanks also to Mark Smucker for help indexing the corpus and Cindy Loiselle for her careful editing suggestions. This work was supported in part by the Center for Intelligent Information Retrieval.

10. REFERENCES

[1] Allan, J., Callan, J., Croft, W. B., Ballesteros, L., Broglio, J., Xu, J., and Shu, H. 1997. INQUERY at TREC-5. In *Proceedings of the 5th Text Retrieval Conference.* NIST, 119–132.

[2] Attardo, S. and Raskin, V. 1991. Script theory revis(it)ed: Joke similarity and joke representation model. *Humor: International Journal of Humor Research* 4(3-4), 293–347.

[3] Bendersky, M. and Croft, W. B. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM Press, New York, NY, 491–498.

[4] Berger, A. and Lafferty, J. 1999. Information retrieval as statistical translation. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM Press, New York, NY, 222–229. DOI= http://dx.doi.org/10.1145/312624.312681

[5] Binsted, K., Bergen, B., Coulson, S., Nijholt, A., Stock, O., Strapparava, C., Ritchie, G., Manurung, R., Pain, H., Waller, A., and O'Mara, D. 2006. Computational humor. *IEEE Intelligent Systems*, 21(2), 59–69. DOI= http://dx.doi.org/10.1109/MIS.2006.22

[6] Brown, P. F., Cocke, J., Della Pietra, S., Della Pietra, V. J., Jelinek, F., Lafferty, J. D., Mercer, R. L., and Roossin, P. S. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2), 79–85.

[7] Goldberg, K., Roeder, T., Gupta, D., and Perkins, C. 2001. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval Journal*, 4(2), 133–151.

[8] Hofstadter, D. and Gabor, L. 1989. Synopsis of the workshop on humor and cognition. *Humor: International Journal of Humor Research*, 2(4), 417–440.

[9] Kruger, A., Giles, C. L., Coetzee, F. M., Glover, E., Flake, G. W., Lawrence, S., and Omlin, C. 2000. DEADLINER: Building a new niche search engine. In *Proceedings of the 9th International Conference on Information and Knowledge Management.* ACM Press, New York, NY, 272–281.

[10] Lafferty, J. and Zhai, C. 2001. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM Press, New York, NY, 111–119. DOI= http://dx.doi.org/10.1145/383952.383970

[11] Manning, C. D., Raghavan, P., and Schütze, H. 2008. *Introduction to Information Retrieval.* Cambridge University Press.

[12] McCallum, A. K., Nigam, K., Rennie, J., and Seymore, K. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2), 127–163. DOI= http://dx.doi.org/10.1023/A:1009953814988

[13] Mihalcea, R. 2007. Multidisciplinary facets of research on humour. In Masulli, F., Mitra, S., and Pasi, G., eds., *Applications of Fuzzy Sets Theory (Proceedings of the Workshop on Cross-Language Information Processing)*, Lecture Notes in Artificial Intelligence. Springer, 412–421.

[14] Mihalcea, R. and Strapparava, C. 2006. Technologies that make you smile: Adding humor to text-based applications. *IEEE Intelligent Systems*, 21(5), 33–39.

[15] Motro, A. 1988. VAGUE: A user interface to relational databases that permits vague queries. *ACM Trans. Inf. Syst.*, 6(3), 187–214.

[16] Raskin, V. 1985. *Semantic Mechanisms of Humor*. Studies in Linguistics and Philosophy. D. Reidel.

[17] Ritchie, G. 2003. *The Linguistic Analysis of Jokes*. Routledge Studies in Linguistics, Vol. 2. Routledge, London.

[18] Schatz, B. R. 2002. The Interspace: Concept navigation across distributed communities. *Computer*, 35, 1 (Jan. 2002), 54–62.

[19] Taylor, J. M. and Mazlack, L. J. 2007. Multiple component computational recognition of children's jokes. In *IEEE International Conference on Systems, Man and Cybernetics.* 1194–1199.

[20] Witten, I. H. and Frank, E. 2005. *Data Mining: Practical Machine Learning Tools and Techniques,* 2nd Edition. Morgan Kaufmann, San Francisco, CA.

[21] Woods, W. A. 1997. Conceptual indexing: A better way to organize knowledge. Technical Report SMLI TR-97-61. Sun Microsystems Laboratories, Mountain View, CA.

[22] Woods, W. A., Bookman, L. A., Houston, A., Kuhns, R. J., Martin, P., and Green, S. 2000. Linguistic knowledge can improve information retrieval. In *Proceedings of the 6th Conference on Applied Natural Language Processing.* Morgan Kaufmann, San Francisco, CA, 262–267. DOI= http://dx.doi.org/10.3115/974147.974183

[23] Zhai, C. and Lafferty, J. 2001. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the 10th International Conference on Information and Knowledge Managemen*t. ACM Press, New York, NY, 403–410. DOI= http://dx.doi.org/10.1145/502585.502654

[24] Zhu, J., Eisenstadt, M., Song, D., and Denham, C. 2006. Exploiting semantic association to answer 'vague queries'. In Li, Y., Looi, M., and Zhong, N., eds., *Advances in Intelligent IT – Active Media Technology 2006*. Frontiers in Artificial Intelligence and Applications, Vol. 138. IOS Press, 73–78.

[25] Zrehen, S. and Arbib, M. A. 1998. Understanding jokes: A neural approach to content-based information retrieval. In *Proceedings of the 2nd International Conference on Autonomous Agents*. ACM Press, New York, NY, 343–349. DOI= http://dx.doi.org/10.1145/280765.280856

[26] "Logic Problems – easy," http://www.folj.com/puzzles/easy.htm

[27] "The Aristocrats (2005)," The Internet Movie Database, http://www.imdb.com/title/tt0436078/

[28] "Brain Teasers and Math Puzzles," Syvum Technologies, http://www.syvum.com/teasers/