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. In particular, we consider the task of identifying the "same joke" to be a necessary component of any joke retrieval system, and we examine it 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 marked up 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 Search and Retrieval**]

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. If you believe Hollywood writers, humor will be the last skill for artificial intelligences to acquire. If you believe linguistic and computational researchers, jokes are a domain where "the question of semantics can no longer avoided" [15]; and worse, they contain "language that requires deep conceptual knowledge about the details of human experience" [4].

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 similar 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—say, to determine if any version of it is present in a given corpus. For instance, with the example above, 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. It is not just the problem of synonymy here, but also that of knowing which aspects of a joke are likely to vary and capturing the wide range of possible alternatives.

Jokes illustrate this structural similarity well, but they are just one of many domains where it is important. A closely related genre 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 [16] shows one immediately, for example, how to use a 70 mL test tube and a 40 mL test tube to measure exactly 60 mL of hydrochloric acid. In fact, every grade school "story problem" probably fits into a small number of templates; it is easy to generate new problems of a given type [18], but fortunately for teachers, no reverse tool (automatic recognition and solution of homework problems) is yet known to us. Unfortunately for researchers, neither is there 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 [10][13].

Another situation where it is difficult to formulate queries to describe a particular meaning is in 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, even though the quote we *wanted* is likely out there. Song lyrics have the same property: one must remember them exactly to find them on the web. A similar domain is that of proverbs: a given saying may be

[1] See, for example, the recent documentary "The Aristocrats," which explores the variations of a single joke [17].

expressed in numerous ways, particularly across cultures, and it would be interesting to find versions of the same message.

> 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.

One final example where structure can matter more than word content is cooking. Websites with recipes can suggest other recipes that have "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 exactly the same steps and the same seasonings as for the chicken with green beans. Retrieving other recipes with the same structure would make it easier to learn which aspects one can vary.

In this paper, we limit ourselves to (studying) jokes, and we consider the question of recognizing whether two documents are "the same joke," or as we will term it, when they belong to the same "joke cluster." This 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.

We consider the task of pairwise classification in Section 4: given two jokes, decide if they match. Next, in Section 5 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 5.3 we incorporate a classifier into our ranking function. We begin now by introducing our data and document models.

2. 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 that fell outside our definition of jokes. 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 Q/A 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 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 these. The clusters range in size from 2 to 13 jokes, and they include a total of 217 jokes. 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]."

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 most representative of future performance will be those on the short jokes. However, it is also possible that the same bias—perhaps, that longer jokes were more interesting to look for, and that shorter jokes were

harder to vary, often just word puns like the "snailor"—would affect the queries of future users.

3. METHODS

We use a language modeling approach. The document models and similarity measures described next are employed in both the classification and ranking tasks.

3.1 Document Models

3.1.1 Baseline

The baseline is a standard unigram (bag of words) model. With this, for a word w and a document d, the initial probability of a word given the document model is the maximum likelihood estimate:

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

Then, linear interpolation smoothing is used 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, separately for each model and task. In the ranking setting, the value of $\lambda = 0.99$ is optimal for all models; for classification, the value $\lambda = 0.4$ is near-optimal for all models.

Throughout this paper, the query is also a document from our collection. However, we do not smooth the query model:

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

3.1.2 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. The same equations above are used, only every document in the corpus is truncated.

3.1.3 Annotations

The second alternative addresses the idea of interchangeable elements in a joke. For example, if one can generally vary characters and locations in jokes, then it may be a helpful abstraction to introduce the tokens "#person" and "#location." And so on with other common categories. In place of plain text, we then have a higher-level representation, like this (this example is shown 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]"

In the best case, the words not annotated would be verbs and other words that convey the generic meaning of the joke. 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 lawyer …"

To implement the annotations, there are two aspects to decide: (a) how to mark up the text with annotations, and (b) how to use them. For the first question, we created word lists for each category (see Table 1). During preprocessing, any document word that matches a list word is marked up. This is a coarse method and yields obvious markup errors, for example with homonyms and irregular plurals, but such problems are present already in the bag of words model.

Table 1. Categories of annotations.

animal	number
color	organization
currency	person
location	timeDate
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 normal word under the annotations model, $P(w|M_d)_{MLE}$ is as before, but for a word w annotated from word list A, the formula changes to this:

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

3.1.4 Using the Markup for Other 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 to the original words.

In our code base, we provided a flexible syntax for specifying document models along the above lines, and we created 108 such variations. The scores from these models are given as inputs to the machine learning classifier introduced below in Section 4.3.

3.2 Similarity Measures

To measure the similarity of a query to a document, we use the Kullback-Leibler (KL) divergence of the query and document models. 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 a constant, as in the retrieval setting, KL divergence is rank-equivalent to the more familiar cross entropy measure $H(p,q)$ [6].

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

The summation above is often shown as 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. We require 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.

4. CLASSIFICATION

In the classification task, we are given two documents and need to 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 are only "learning" a cutoff threshold. There are positive and negative examples, the positives being joke pairs that match, and the negatives being joke pairs that do not match.

4.1 Training and Test Sets

The samples are created in ten groups, to allow ten-fold cross validation. In order that the training and testing barrier be kept intact, each joke cluster only contributes examples to one group. We avoided letting any one large cluster dominate the examples, by using no more than 15 positives and 15 negatives per 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 sampled negatives so that they would be comparable in their *ranks* to the positives. That is, for each positive pair, we took one joke as a query, retrieved a ranked list of jokes, and recorded the rank (in that list) of the second joke. By repeating this with every joke as the query, we sampled a distribution of ranks of positives. Then, to generate negatives, we took one joke from the cluster, retrieved a ranked list of jokes, picked a desired rank from our distribution, and sampled a *non-matching* joke from at or near that rank. This way we created negative examples that were, in theory, difficult to distinguish from the positives.

4.2 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 use a symmetric score. We make the score symmetric by taking the average of both directions, i.e., using $\frac{1}{2}(KL(M_a \,||\, M_b) + KL(M_b \,||\, M_a))$.

It would have been possible to use the symmetric cross entropy instead. Since the actual distribution of scores matters to us—the values, not just the rankings—we chose 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.

4.3 Experiments

In total, we have approximately 600 data points, of which 58% were negatives. We measure the accuracy—the number of correct predictions—for each fold, and then compute an average across the folds. During the training phase, the classifier computes the score for each pair, and chooses a prediction threshold to maximize the accuracy on the training data. 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 notice are that the accuracies are fairly good, and that the models that use joke structures have some advantage over the baseline. Also, there is diversity among the models; Table 3 shows that each 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 scores from the models as inputs to a classifier, and allow the classifier to make the prediction. We use Weka's logistic regression tool [12]; its other classifiers performed similarly or worse. We test several combinations of features: first, the three models we have seen above. Next, with the idea that relative document length would be predictive, we add two features concerning that. Finally, we use as our features the scores from all 108 model variations described in Section 3.1.4.

The results of the classifiers are shown in Table 4. We see that using the set of three features, the classifier achieves better performance than any of the models alone. Adding additional features does not help; if anything, it was useful to manually select the set of three features. We assessed 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.

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

It is surprising in light of Section 5 below, and somewhat misleading, that the punch line model would perform so well in classification. Further analysis shows that for the baseline model, there is not a large separation between the scores of its positive and negative classes. This is a result of the sampling procedure: by intent, the two classes were close in baseline scores. The annotations model has a similar situation. However, the punch line model tends to score differently 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.

5. 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."

5.1 Setup

In this setting, we use one joke as a query, and perform a retrieval using one of the document models described earlier. The relevant documents for this query are 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. We do this for every cluster, and finally we report the averages across all 60 clusters.

5.2 Results

The results of the ranking experiments are displayed in Table 5. We see that unlike in classification, here the baseline model performs best and the punch line model worst. The differences between the baseline and annotations model are 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 3. 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 how each of the alternative models has clusters for which they soundly beat the baseline. This suggests that again there is potential for improvement by combining the scores of the three models.

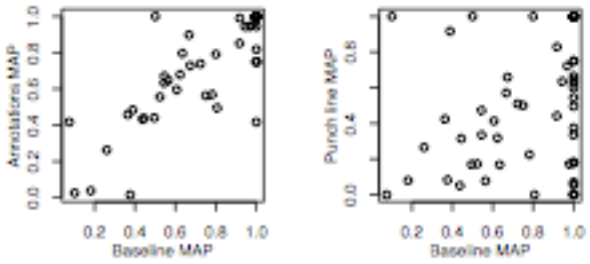

Figure 3. Mean average precision of each joke cluster (one data point per cluster). Left, baseline model versus annotations. Right, baseline versus punch line.

5.3 Re-ranking

To combine the models, we return to the approach from above: training a pairwise classifier using scores from the three models. The classifier actually outputs a probability score, not just a binary decision, so we can take and use this score for ranking. In order to bring this classifier into the ranking setting, for which 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 these 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 4 the recall curve as a function of the number of documents. The curve levels off by 500 documents, at recall = 99.8%.

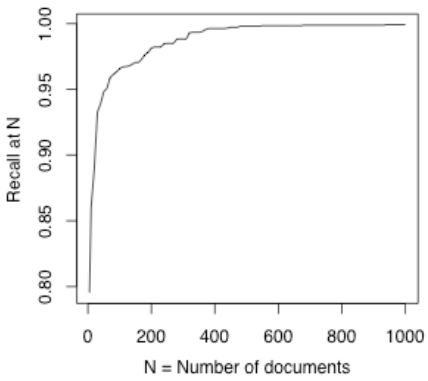

Figure 4. Recall of the baseline model, averaged over all jokes.

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 pairs of jokes in all clusters, for 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 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. We use a ratio of about 1:2 for positives to negatives, which keeps the size of the training set reasonable.)

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 queries in the other 9 groups.

Table 6 shows the results of using the classifier to re-rank the top 500 documents. Using the classifier by itself, the scores are in fact worse than the baseline. Once more, we examine the scatterplot of scores (Figure 5, left). 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 classifiers.

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 right side of Figure 5 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

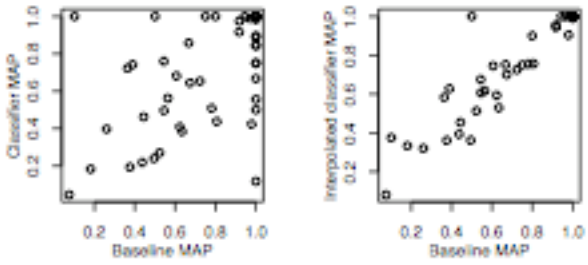

Figure 5. Mean average precision of each joke cluster (one data point per cluster). Left, baseline model versus classifier. Right, baseline versus interpolated classifier.

We performed a few experiments analyzing the contribution of the classifier, and in particular, whether the improvement in score could be achieved in some simpler way. These experiments are shown below, in Table 7. First, one method for improving retrieval in many situations is to expand the query using pseudo-relevance feedback. We used linear interpolation between the original query and the top t documents [14]. 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

6. DISCUSSION

We gain some insight into the utility of the three document models by looking at specific queries where they performed differently.

For the most part, it seems that if a joke is sufficiently long, certain words actually do show up in all its versions. In the challenging-looking joke cluster from Figures 1-3, for example, the baseline model gives a reasonable MAP of 0.62; annotations scores mildly higher.

When a joke is short, the baseline model still performs well if there are distinctive words that appear in every version. For instance, the unusual words "trampoline" and "tire gauge" in the joke versions in Figure 6 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 6. Joke clusters easy for the baseline.

There is a mild indication that joke length correlates with the success of the annotations model. In particular, when 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 3.1.3 involving "skid marks;" in cases like this, there are not always enough words for the baseline model to latch on to. In particular, in the example in Figure 7, the annotations model scored perfectly, whereas the baseline only had a MAP of 0.5.

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 7. Joke clusters easy for annotations, difficult for the baseline.

As for punch lines, when the punch lines match closely, this is 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.

7. RELATED AND FUTURE WORK

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. [4] collects the work of several groups that publish in this field, and a recent review by Mihalcea [8] surveys theories of humor from psychology, philosophy, and other fields. Some recognition tasks include using text classification to distinguish humorous one-liners from other sentences like headlines or proverbs [9], or recognizing short children's jokes using any of several theories of humor [11]. In humor generation, there are systems that generate riddles with puns or humorous acronyms, and then there are systems that insert humor into emails or chatbots as a way to improve human-computer interactions [4], [9]. In addition, there is a jokes search engine called Jester, but it has been created and studied exclusively as a recommender system [6]. The models of humor that help computers recognize or construct it could be valuable to anyone studying jokes, but none of the above work consider variations of a single joke, the central idea of this paper.

The work most similar in spirit to this is an article by Zrehen and Arbib [15]. Critiquing IR techniques as relying too heavily on the specifc words in a document, they propose an architecture for neural networks that would infer the implied context of a sentence and then recognize jokes by the incongruities they contain. This system was not actually implemented, and its task would have again been humor recognition; however, the authors do discuss joke retrieval as a domain where the query words may not be found in a document, and where one must then include semantics in search.

In terms of other possible methods for recognizing joke variants, we considered viewing variants like translations into other languages, and learning a translation model of common word substitutions [5]. 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 [3]. 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 more recently adapted to long queries [2]. It is not clear that concepts which are key in standard text—e.g., proper nouns—would play the same role in jokes; however, it might be possible to modify techniques such as these to work for jokes.

In the example domains we have described where search is difficult because words may change (jokes, puzzles, quotes, etc.), it was still possible to formulate a query that at least paraphrases the ideal document. An even harder problem would be situations where the user can describe what they want, but not in terms that would appear in the document. For instance, "that popular song with the catchy rhythm," or other cases where it is difficult to imitate the document. This kind of question motivates approaches that use semantics and ontologies to try to understand the user [10][13].

8. CONCLUSIONS

We have used knowledge of a particular domain to build a retrieval system that performs better at ranking and classification than the standard model in that domain. Along the way, we have used this domain, humor, to argue for alternative definitions of similarity between documents: that they exist and that they matter. In particular, that documents in certain domains are difficult to search for because one cannot predict the words the item will contain; only their relationships count. To discern such other types of connections may require that information retrieval move beyond the word level and towards understanding meaning, perhaps via modeling the structure of text in various domains.

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" idea. Thanks also to Mark Smucker for his help indexing the corpus. This work was supported in part by the Center for Intelligent Information Retrieval.

10. REFERENCES

[1] James Allan, Jamie Callan, W. Bruce Croft, Lisa Ballesteros, John Broglio, Jinxi Xu, and Hongmin Shu. INQUERY at TREC-5. pages 119–132. NIST, 1997.

[2] Bendersky, M. and Croft, W. B. 2008. Discovering key concepts in verbose queries. In Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '08. ACM Press, New York, NY.

[3] 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. SIGIR '99. ACM Press, New York, NY, 222-229. DOI= http://dx.doi.org/10.1145/312624.312681

[4] 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

[5] 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.

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

[7] 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. SIGIR '01. ACM Press, New York, NY, 111-119. DOI=http://dx.doi.org/10.1145/383952.383970

[8] 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.

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

[10] Schatz, B. R. 2002. The Interspace: concept navigation across distributed communities. Computer, 35, 1 (Jan. 2002), 54-62.

[11] 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. (SMC '07). 1194-1199.

[12] Witten, I. H. and Frank, E. 2005. Data Mining: Practical machine learning tools and techniques, 2nd Edition. Morgan Kaufmann, San Francisco.

[13] Zeng, J. and Yang, Y. 2003. Information retrieval based on conceptual network. In Proceedings of the International Conference on Natural Language Processing and Knowledge Engineering. 380-387.

[14] Zhai, C. and Lafferty, J. 2001. Model-based feedback in the language modeling approach to information retrieval. In Proceedings of the tenth international conference on Information and knowledge management. CIKM '01. ACM Press, New York, NY, 403-410. DOI= http://dx.doi.org/10.1145/502585.502654

[15] 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. AGENTS '98. ACM Press, New York, NY, 343-349. DOI=http://dx.doi.org/10.1145/280765.280856

[16] http://www.folj.com/puzzles/easy.htm

[17] http://www.imdb.com/title/tt0436078/

[18] http://www.syvum.com/teasers/