Generating Synthetic Comparable Questions for News Articles

Oleg Rokhlenko
Yahoo! Research
Haifa 31905, Israel
olegro@yahoo-inc.com

Idan Szpektor
Yahoo! Research
Haifa 31905, Israel
idan@yahoo-inc.com

Abstract

We introduce the novel task of automatically generating questions that are relevant to a text but do not appear in it. One motivating example of its application is for increasing user engagement around news articles by suggesting relevant comparable questions, such as "*is Beyonce a better singer than Madonna?*", for the user to answer. We present the first algorithm for the task, which consists of: (a) offline construction of a comparable question template database; (b) ranking of relevant templates to a given article; and (c) instantiation of templates only with entities in the article whose comparison under the template's relation makes sense. We tested the suggestions generated by our algorithm via a Mechanical Turk experiment, which showed a significant improvement over the strongest baseline of more than 45% in all metrics.

1 Introduction

For companies whose revenues are mainly ad-based, *e.g.* Facebook, Google and Yahoo, increasing user engagement is an important goal, leading to more time spent on site and consequently to increased exposure to ads. Examples for typical engaging content include other articles for the user to read, updates from the user's social neighborhood and votes or comments on videos, blogs etc.

In this paper we propose a new way to increase user engagement around news articles, namely suggesting questions for the user to answer, which are related to the viewed article. Our motivation is that there are questions that are "irresistible" because they are fun, involve emotional reaction and expect simple answers. These are comparative questions, such as "*is Beyonce a better singer than Madonna?*", "*who is better looking, Brad Pitt or George Clooney?*", "*who is faster: Superman or Flash?*" and "*which camera brand do you prefer: Canon or Nikon?*" Furthermore, such questions are social in nature since users would be interested in reading the opinions of other users, similar to viewing other comments (Schuth et al., 2007). Hence, a user that provided an answer may return to view other answers, further increasing her engagement with the site.

One approach for generating comparable questions would be to employ traditional question generation, which syntactically transform assertions in a given text into questions (Mitkov et al., 2006; Heilman and Smith, 2010; Rus et al., 2010). Sadly, fun and engaging comparative questions are typically not found within the text of news articles. A different approach would be to find concrete relevant questions within external collections of manually generated comparable questions. Such collections include Community-based Question Answering (CQA) sites such as Yahoo! Answers and Baidu Zhidao and sites that are specialized in polls, such as Toluna. However, it is highly unlikely that such sources will contain enough relevant questions for any news article due to typical sparseness issues as well as differences in interests between askers in CQA sites and news reporters. To better address the motivating application above, we propose the novel task of automatically suggesting comparative questions that are relevant to a given input news article but do not appear in it.

To achieve broad coverage for our task, we present an algorithm that generates synthetic concrete questions from *question templates*, such as "*Who is a better actor: #1 or #2?*". Our algorithm consists of two parts. An offline part constructs a database of comparative question templates that appear in a large question corpus. For a given news article, an online part chooses relevant tem-

Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, pages 742–751,
Sofia, Bulgaria, August 4-9 2013.

Who's Hollywood's Hottest Dad?

In honor of Father's Day, Us Weekly wants to know -- who do you think is the hottest dad in Hollywood?

David Beckham, 36 -- who's expecting baby No. 4 with wife **Victoria** -- is already father to sons Brooklyn, 12, Romeo, 8, and Cruz, 6; **Brad Pitt**, 47, co-parents six children with **Angelina Jolie** (Maddox, 9, Pax, 7, Zahara, 6, Shiloh, 5, and twins Knox and Vivienne, 2); **Will Smith** helped pave the way for his superstar children Jaden, 12, and Willow, 10; and **Matthew McConaughey**, 41, is the proud father of cuties Levi, 2, and Vida, 1.

Tell **Us**: Who's the hottest dad in Hollywood?

Figure 1: An example news article from OMG!

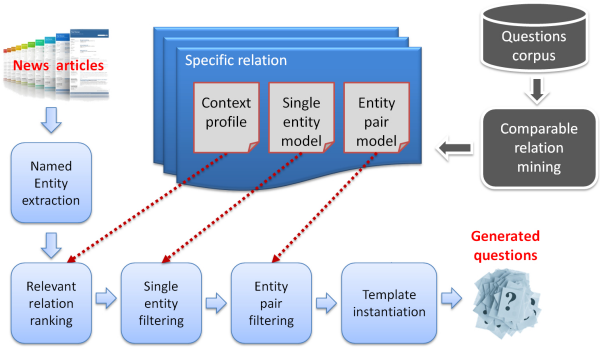

Figure 2: A high-level overview of the comparable question generation algorithm. The offline part is colored dark grey and the online part is colored light blue.

plates for the article by matching between the article content and typical template contexts. The algorithm then instantiates each relevant template with two entities that appear in the article. Yet, for a given template, only some of the entities are plausible slot fillers. For example, '*Madonna*' is not a reasonable filler for "*Who is a better dad, #1 or #2?*". Thus, our algorithm employs entity filtering to exclude candidate instantiations that do not make sense.

To test the performance of our algorithm, we conducted a Mechanical Turk experiment that assessed the quality of suggested questions for news articles on celebrities. We compared our algorithm to a random baseline and to a partial version of our algorithm that includes a template relevance component but lacks filtering of candidate instantiations. The results show that the full algorithm provided 45% more correct instantiations, but surprisingly also 46% more relevant suggestions compared to the stronger baseline. These results point at the importance of both picking relevant templates and smart instantiation selection to the quality of generated questions. In addition, they indicate that user perception of relevance is affected by the correctness of the question.

2 Motivation and Algorithmic Overview

Before we detail our algorithm, we provide some motivations and insights to the design choices we took in our algorithm, which also indicate the difficulties inherent in the task.

2.1 Motivation

Given a news article, our algorithm generates a set of comparable questions for the article from question templates, *e.g.* "*who is faster #1 or #2?*". Though the template words typically do not appear in the article, they need to be *relevant* to it's content, that is they should correspond to one of the main themes in the article or to one of the public interests of the compared entities. For example, "*who is a better dad #1 or #2?*" is relevant to the article in Figure 1, while "*who is faster #1 or #2?*" is not relevant. Therefore, we need to model the typical contents to which each template is relevant.

Looking at the structure of comparable questions, we observed that a specific *comparable relation*, such as '*better dad*' and '*faster*', can usually be combined with named entities in several syntactic ways to construct a concrete question. We encode this information in *generic comparable templates*, *e.g.* "*who is a RE: #1 or #2?*" and "*is #1 a RE than #2?*", where *RE* is a slot for a comparable relation and *#1* and *#2* are slots for entities. Using the above generic templates, '*Jet Li*' and '*Jackie Chan*' can be combined with the comparable relation '*better fighter*' to generate "*who is a better fighter: Jackie Chan or Jet Li?*" and "*is Jackie Chan a better fighter than Jet Li?*" respectively. Following, our algorithm separately maintains comparable relations and generic templates.

In this paper we constrain ourselves to generate comparable questions between entities that appear in the article. Yet, not all entities can be compared to each other under a specific template, adding substantial complexity to the generation of questions. Looking at Figure 1, the generated question '*who is faster, Angelina Jolie or David Beckham?*' makes sense with respect to *David Beckham*, but not with respect to *Angelina Jolie*, since the typical reader is rarely interested in her running skills. Our algorithm thus needs to assess whether an instantiation is *correct*, that is whether the comparison between the two entities makes sense under the specific template.

Further delving into question correctness, the above example shows the need to assess each entity by itself. However, even if both entities are independently valid for the template, their comparison may not make sense. For example, "*who is better looking: Will Smith or Angelina Jolie?*" doesn't feel right, even though each entity by itself fits the template. This is because when comparing looks, we expect a same sex comparison.

2.2 Algorithmic Overview

The above observations led us to the design of the automatic generation algorithm depicted in Figure 2. The algorithm's offline part constructs, from a large collection of questions, a database of comparable relations, together with their typical contexts. It also extracts generic templates and the mapping to the relations that may instantiate them. From this database, we learn: (a) a *context profile* per template for relevance matching; (b) a *single entity model* per template slot that identify valid instantiations; and (c) an *entity pair model* that detects pairs of entities that can be compared together under the template. In the online part, these three models are applied to rank relevant templates for a given article and to generate only correct questions with respect to template instantiation.

The next two sections detail the template extraction component and the model training and application component in our algorithm.

3 Comparable Question Mining

To suggest comparable questions our algorithm needs a database of question templates. As discussed previously, a good source for mining such templates are CQA sites. Specifically, in this study we utilize all questions submitted to Yahoo! Answers in 2011 as our corpus. We next describe how comparable relations and generic comparable templates are extracted from this corpus.

3.1 Comparable Relation Extraction

An important observation for the task of comparable relation extraction is that many relations are complex multiword expressions, and thus their automatic detection is not trivial. Examples for such relations are marked in the questions "*Who is the best rapper alive, Eminem or Jay-z?*" and "*Who is the most beautiful woman in the world, Adriana Lima or Jessica Alba?*". Therefore, we decided to employ a Conditional Random Fields (CRF) tagger (Lafferty et al., 2001) to the task, since CRF was shown to be state-of-the-art for sequential relation extraction (Mooney and Bunescu, 2005; Culotta et al., 2006; Jindal and Liu, 2006).

As a pre-processing step for detecting comparable relations, our extraction algorithm identifies all the named entities of interest in our corpus, keeping only questions that contain at least two entities. In each of remaining questions, we then substitute the entity names with the variable slots #i in the order of their appearance. For example, "*Nnamdi Asomugha vs. Darrelle Revis? Who is the better cornerback?*" turned into "*#1 vs. #2? Who is the better cornerback?*". This transformation helps us to design a simpler CRF than that of (Jindal and Liu, 2006), since our CRF utilizes the known positions of the target entities in the text.

To train the CRF model, the authors manually tagged all comparable relation words in approximately 300 transformed questions in the filtered corpus. The local and global features for the CRF, which we induce from each question word, are specified in Figures 3 and 4 respectively. Though there are many questions in Yahoo! Answers containing two named entities, *e.g.* "*Is #1 dating #2?*", our CRF tagger is trained to detect only comparable relations like "*Who is prettier #1 or #2?*". This is due to the labeled training set, which contains only this kind of relations, and to our features, which capture aspects of this specific linguistic structure.

The trained model was then applied to all other questions in the filtered corpus. This tagging process resulted in 60,000 identified question relation occurrences. From this output we constructed a database consisting of all occurring relations; each relation is accompanied by its *supporting questions*, those questions in which the relation occurrences were found. To achieve a highly accurate database, we filtered out relations with less than 50 supporting questions, ending with 295 relations in our database[1]. The authors conducted a manual evaluation of the CRF tagger performance, which showed 80% precision per occurrence. Yet, our filtering above of relations with low support left us with virtually 100% precision per relation and per occurrence.

[1] We intend to make this database publicly available under Yahoo! Webscope™ (`http://webscope.sandbox.yahoo.com`).

[2] `http://nlp.stanford.edu/software/`

(a) The word itself
(b) Whether the word is capitalized
(c) The word's suffixes of length 1,2, and 3, which helps in detecting comparative adjectives that ends '*est*' or '*er*'
(d) The word's position in the sentence
(e) The word's Part of speech (POS) tag, based on the Stanford POS tagger[2]
(f) The words in a window of ± 3 around the current one
(g) The adjective before the word, if exists, which helps detecting comparative noun phrases, *e.g.* '*better driver*' and '*best singer*'
(h) The shortest word distance between the word and one of the #i variables.
(i) The shortest word distance of the word to one of the following connectives: '*between*', '*out*', '*:*', '*,*', '*?*'

Figure 3: CRF local features for each word

(a) WH question type of the question, *e.g. what*, *which*, *who*, *where*
(b) The average word distance between all #i variables in the question
(c) The conjunction tokens appearing between the #i variables, such as *or*, *vs*, *and*

Figure 4: CRF global features for each word

3.2 Comparable Template Extraction

Our second mining task is to extract generic comparable templates that appear in our corpus, as well as identifying which comparable relation can instantiate which generic template.

To this end, we replace each recognized relation sequence with a variable *RE* in the support questions annotated with #i variables. For example, "*who is the best rapper alive, #1 or #2?*" is transformed to "*who is RE, #1 or #2?*". We next count the occurrences of each templatized question. While some questions contain many details besides the comparable generic template, others are simpler and contain only the generic template. Through this counting, frequently occurring generic templates are revealed, such as "*is #1 a RE than #2?*". We retain only generic templates which appeared more than 50 times.

Finally, for each comparable relation we mark as applicable only generic templates that occur at least once in the supporting questions of this relation. For example, the template "*who is RE: #1 or #2?*" was found applicable for '*funnier*', and thus could be used to generate the concrete question "*who is funnier: Jennifer Aniston or Courteney Cox?*". On average, each relation was associated with 3 generic templates.

Algorithm 1 A high level overview of the online part of the question generation algorithm

Input: A news article
Output: A sorted list of comparable questions
1: Identify all target named entities (NEs) in the article
2: Infer the distribution of LDA topics for the article
3: For each comparable relation R in the database, compute its relevance score to be the similarity between the topic distributions of R and the article
4: Rank all the relations according to their relevance score and pick the top M as relevant
5: **for** each relevant relation R in the order of relevance ranking **do**
6: Filter out all the target NEs that do not pass the single entity classifier for R
7: Generate all possible NE pairs from the those that passed the single classifier
8: Filter out all the generated NE pairs that do not pass the entity pair classifier for R
9: Pick up the top N pairs with positive classification score to be qualified for generation
10: Instantiate R with each chosen NE pair via a randomly selected generic template
11: **end for**

4 Online Question Generation

The online part of our automatic generation algorithm takes as input a news article and generates concrete comparable questions for it. Its high level description is presented in Algorithm 1. The algorithm starts with identifying the comparable relations in our database that are relevant to the article. For each relevant relation, we then generate concrete questions by picking generic templates that are applicable for this relation and instantiating them with pairs of named entities appearing in the article. Yet, as discussed before, only for some entity pairs the comparison under the specific relation makes sense, a quality which we refer to as instantiation correctness (see Section 2). To this end, we utilize two supervised models to filter incorrect instantiations. We next detail the two aspects of the online part: ranking relevant relations and correctly instantiating relations.

4.1 Ranking relevant relations

To assess how relevant a given comparable relation is to an article, we model the relation's typical context as a distribution over latent semantic topics. Specifically, we utilize Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to infer latent topics in texts.

To train an LDA model, we constructed for each comparable relation a pseudo-document consisting of all questions that contain this relation in our corpus (the supporting questions). We then

trained a model of 200 topics over these pseudo-documents, resulting in a model over a lexicon of 107,835 words. An additional product of the LDA training process is a topic distribution for each relation's pseudo-document, which we consider as the relation's *context profile*. We note that, unless otherwise specified, different model parameters were chosen based on a small held out collection of articles and questions, manually annotated by the authors. This collection was used to validate that the chosen parameter values indeed "make sense" for the task.

Given a news article, a distribution over LDA topics is inferred from the article's text using the trained model. Then, a cosine similarity between this distribution and the context profile of each comparable relation in our database is computed and taken as the relevance score for this relation. Finally, we rank all relations according to their relevance score and pick the top M as candidates for instantiation (M=3 in our experiment).

4.2 Correctly instantiating relations

To generate useful questions from relevant comparable relations, we need to retain only correct instantiations of these relations. To this end, we utilize two complementing types of filters, one for each entity by itself, and one for pairs, since each filter considers different attributes of the entities at hand. For example, for the relation '*is faster*', the single entity filter looks for athletes of all kinds, for whom this comparison is of interest to the reader. The pair filter, on the other hand, attempts to pass only same sex and same profession comparisons, *e.g.* male football players or female baseball players for this relation.

We next describe the various features we extract for every entity and the supervised models that given this feature vector representation assess the correctness of an instantiation.

4.2.1 Entity Features

We want to represent each entity as a vector of features that capture different aspects of entity characterization. To this end, we utilize two different broad-scale sources of information about named entities. The first is DBPedia[3], which contains structured information on entries in Wikipedia, many of them are named entities that appear in news articles. The second source is the corpus of CQA questions, which in our study was harvested from Yahoo! Answers (see Section 3).

For named entities with a DBPedia entry, we extract all the DBPedia properties of classes *subject* and *type* as indicator features. Some example features for *Brad Pitt* include *Actors_from_Oklahoma*, *AmericanAtheists*, *Artist* and *American_film_producers*.

One property that is currently missing from DBPedia is *gender*, a feature that was found to be very useful in our experiments. We automatically induce this feature from the Wikipedia abstract in each DBPedia entry. Specifically, we construct a histogram of male and female pronouns: *he* and *his* vs. *she* and *her*. The majority pronoun sex is then chosen to be the gender of the named entity, or none if the histogram is empty.

One way to utilize the CQA question corpus could be to extract co-occurring words with each target entity as relevant contexts. Yet, since our questions come from Yahoo! Answers, we decided to use another attribute of the questions, the category to which the question is assigned, within a hierarchy of 1,669 categories (*e.g.* '*Sports>Baseball*' and '*Pets>Dogs*'). For each named entity, we construct a histogram of the number of questions containing it that are assigned to each category. This histogram is normalized into a probability distribution with Laplace smoothing of 0.03, to incorporate the uncertainty that lies in named entities that appear only very few times. The categories and their probabilities are added as features, providing a high level representation of relevant contexts for the entity.

4.2.2 Single entity filtering

We view the task of single entity filtering as a classification task. To this end, we trained a classifier per relation, constructing a different labeled training set for each relation. Positive examples are the entities that instantiate this relation in our CQA corpus. As negative examples, we take named entities that were never seen instantiating the relation in the corpus, but still occurred in some questions. We note that our named entity tagger could recognize more than 200,000 named entities, and most of them are negative for a given relation.

For each relation we select negative examples by sampling uniformly from its negative entity list, assuming that the probability of hitting false negatives is low for such a long list. It is known that better classification performance is typically

[3] http://wiki.dbpedia.org/About

achieved for a balanced training set (Provost, 2000). In our case, we over sample to help the classifier explore the large space of negative examples. Specifically, we sample 2,000 negative examples and duplicate the positive set to reach a similar number.

We utilize the Support Vector Machines (SVM) implementation of LIBSVM (Chang and Lin, 2011) with a linear kernel as our classifier. The feature vector of each named entity was induced as described in Section 4.2.1. We split the labeled dataset into 70% training set and 30% validation set. Feature selection using information gain was performed on the training set to throw out non-significant features (Mitchell, 1997). The average accuracy of the single classifiers, measured over the validation sets, was 91%.

4.2.3 Entity pair filtering

Similar to single entity filtering, we view the task of filtering entity pairs as a classification task, training a separate classifier for each relation. Entity pairs that instantiate the given relation in the question corpus are considered positive examples. Yet, the space of all the pairs that never instantiated the relation is huge, and the set of positive examples is relatively much smaller compared to the situation in the single entity classifier. In our study, uniform negative example sampling turned the training into a trivial task, preventing from the classifier to choose an useful discriminative boundary. Therefore, we generate negative examples by sampling only from pairs of named entities that both pass the single entity filter for this relation. The risk here is that we may sample false negative examples. Still, this sampling scheme enabled the classifier to identify better discriminative features.

To generate features for a candidate pair, we take the two feature vectors of the two entities and induce families of pair features by comparing between the two vectors. Figure 5 describes the various features we generate. We utilize LIBSVM with an RBF kernel for this task, splitting the examples into 70% training set and 30% validation set. We over sampled the positive examples to reach up to 100 examples.

The average accuracy of the pair classifiers on the validation set was 83%. For example, named entities that pass the single entity filtering for "*be funny*", include Jay Leno, David Letterman (American TV hosts), Jim Carrey, and Steve Martin (actors). The pair classifier assigned positive scores only to {*Jay Leno*, *David Letterman*} (TV hosts) and {*Jim Carrey*, *Steve Martin*} (actors) but not to other pairings of these entities.

(a) All shared DBPedia indicator features in the two vectors: $f_a^{DBPedia} \cap f_b^{DBPedia}$, indicating them as shared, *e.g.* '*FilmMaker_s*'
(b) All DBPedia features that appear only in one of the vectors, termed *one-side features*: $f_a^{DBPedia} \setminus f_b^{DBPedia}$ and $f_b^{DBPedia} \setminus f_a^{DBPedia}$, indicating them as such, *e.g.* '*FilmMaker_o*'
(c) Wikipedia categories that are ancestors of at least two one-side features that appear in the training set. For example, a common ancestor of '*Spanish_actors*' and '*Russian_actors*' is '*European_actors*'. These features provide a high level perspective on one-side features
(d) The Yahoo! Answers categories in which both named entities appear
(e) Hellinger distance (Pollard, 2001) between the probability distributions over categories of the two entities
(f) Three indicator gender features: whether both named entities are males, both are females or are different

Figure 5: The entity pair features generated from two single entity feature vectors f_a and f_b

5 Evaluation

5.1 Experimental Settings

To evaluate our algorithm's performance, we designed a Mechanical Turk (MTurk) experiment in which human annotators assess the quality of the questions that our algorithm generates for a sample of news articles. As the source of test articles, we chose the OMG! website[4], which contains news articles on celebrities.

Test articles were selected by first randomly sampling 5,000 news article from those that were posted on OMG! in 2011. We then filtered out articles that are longer than 4,000 characters, which were found to be tiresome for annotators to read, and those that are shorter than 300 characters, which consist mainly of video and photos. We were left with a pool of 1,016 articles from which we randomly sampled 100 as the test set.

For each test article our algorithm obtained the top three relevant comparable relations, and for each relation selected the best instantiation (if exists). We used two baselines for performance comparison. The first *random baseline* chooses a relation randomly out of all possible relations in the database and then instantiates it with a random pair of entities that appear in the article. The second *relevance baseline* chooses the most relevant

[4] http://www.omg.com/

	Relevance	Correctness
Random baseline	29%	43%
Relevance baseline	37%	53%
Full algorithm	54%	77%

Table 1: Relevance and correctness percentage by tested algorithm

relation to the article based on our algorithm, but still instantiates it with a random pair. For each test article, we presented to the evaluators the questions generated by the three tested algorithms in a random order to avoid any bias. We note that our second baseline enabled us to measure the stand-alone contribution of the LDA-based relevance model. In addition, it enabled us to measure the relative contribution of the instantiation models on top of relevance model.

Each article was evaluated by 10 MTurk workers, which were asked to mark for each displayed question whether it is relevant and whether it is correct (see Section 2 for relevance and correctness definitions). The workers were given precise instructions along with examples before they started the test. A control story was used to filter out dishonest or incapable workers[5].

5.2 Results

For each tested algorithm, we separately counted the percentage of annotations that marked each question as relevant and the percentage of annotations that marked each question as instantiated correctly, denoted *relevance score* and *correctness score*. We then averaged these scores over all questions that were displayed for the test articles. The results are presented in Table 1. The differences between the full algorithm and the baselines are statistically significant at $p < 0.01$ and between baselines the differences are statistically significant at $p < 0.05$ using the Wilcoxon double-sided signed-ranks test (Wilcoxon, 1945).

Our main result is that our full algorithm substantially outperforms the stronger relevance baseline. It improves the correctness score by 45%, which points at the effectiveness of our two step filtering of incorrect instantiations. It's performance is just under 80%, showing high quality entity pair selection for relations. Yet, we did not expect to see an increase of 46% in the relevance metric, since both the full algorithm and the relevance baseline use the same relevance component to rank relations by. One explanation for this is that sometimes the instantiation filter eliminates all possible entity pairs for some relation that is incorrectly considered relevant by the algorithm. Thus, the filtering of entities provides also an additional filtering perspective on relevance. In addition, it may be that humans tend to be more permissive when assessing the relevance of a correctly instantiated question.

To illustrate the differences between baselines and the full algorithm, Table 2 presents an example article together with the suggested generated questions by each algorithm. The random baseline picked an irrelevant relation, and while the relevance baseline selected a relevant relation, "*a better president*", it was instantiated incorrectly. The full algorithm, on the other hand, both chose relevant relations for all three questions and instantiated them correctly. Especially, the incorrectly instantiated relation in the relevance baseline is now correctly instantiated with plausible presidential candidates.

Comparing between baselines, the relevance baseline beats the random baseline by 28% in terms of relevance. This is not surprising, since this was the focus of this baseline. Yet, it also improved correctness by 23% over the random baseline. This is an unexpected result that indicates that when users view relevant relations, they may be more forgiving in their perception of unreasonable instantiations.

For each article, our full algorithm attempts to generate three questions, one for each of the top three relevant questions. It is possible that for some articles not all three questions will be generated, due to instantiation filtering. We found that for 85% of the articles all three questions were generated. For the remaining 15% at least one question was always generated, and for $\frac{1}{3}$ of them two questions were composed. Furthermore, we found that the relevance and correctness scores were not affected by the position of the question. In the case of instantiation correctness, since the best pair was picked for each relation and this component is quite accurate, this is somewhat expected. In the case of relevance, this indicates that there are usually several relations in our database that are relevant to the article.

[5]We intend to make the tested articles, the instructions to annotators and their annotations publicly available under Yahoo! Webscope™ (`http://webscope.sandbox.yahoo.com`).

Ron Livingston is teaming up with Tom Hanks and HBO again after their successful 2001 collaboration on Band of Brothers. The actor has been cast in HBO's upcoming film Game Change that centers on the 2008 presidential campaign, Deadline reports. He joins Ed Harris, Julianne Moore and Woody Harrelson. The Jay Roach-directed movie follows John McCain (Harris) as he selects Alaska Gov. Sarah Palin (Moore) as his running mate, throughout the campaign and to their ultimate defeat to Barack Obama. Livingston will play Mark Wallace, one of the campaign's senior advisors and the man who prepped Palin for her debate. Harrelson will play campaign strategist Steve Schmidt...

Algorithm	Question
Random baseline	Who is a better singer, Sarah Palin or Barack Obama ?
Relevance baseline	Would Ron Livingston be a better president than Julianne Moore ?
Full algorithm	Who has the best movies Tom Hanks or Julianne Moore ?
Full algorithm	Is John Mccain a better leader than Barack Obama ?
Full algorithm	Would Sarah Palin be a better president than John Mccain ?

Table 2: Automatically generated questions by the baselines and the full algorithm to an example article

5.3 Error Analysis

To better understand the performance of our algorithm, we looked at some low quality questions that were generated, either due to incorrect instantiation or due to irrelevance to the article.

Starting with relevance, one of the repeating mistakes was promoting relations that are related to a list of named entities in the article, but not to its main theme. For example, the relation '*who is a better actor*' was incorrectly ranked high for an article about Ricky Gervais claiming that he has been asked to host Globes again after he offended Angelina Jolie, Johnny Depp, Robert Downey Jr. and Charlie Sheen, among others during last Globes ceremony. The reason for this mistake is that many named entities appear as frequent terms in LDA topics, and thus mentioning many names that belong to a single topic drives LDA to assign this topic a high probability. Yet, unlike other cases, here entity filtering does not help ignoring such errors, since the same entities that triggered the ranking of the relation are also valid instantiations for it.

Analyzing incorrect instantiations, many mistakes are due to mismatches between the two compared entities that were too fine grained for our algorithm to catch. For example, "*who's the better guitarist: Paul McCartney or Ringo Starr?*" was generated since our algorithm failed to identify that *Ringo Starr* is a drummer rather than a guitarist, though both participants in the relation are musicians. In other cases, strong co-occurrence of the two celebs in our question corpus convinced the classifiers that they can be matched. For example, "*who is a better dancer Michael Jackson or Debbie Rowe?*" was incorrectly generated, since Debbie Rowe is not a dancer. Yet, she was Michael Jackson's wife and they appear together in a lot of questions in our corpus.

6 Related Work

Traditionally, question generation focuses on converting assertions in a text into question forms (Brown et al., 2005; Mitkov et al., 2006; Myller, 2007; Heilman and Smith, 2010; Rus et al., 2010; Agarwal et al., 2011; Olney et al., 2012). To the best of our knowledge, there is no prior work on our task, which is to generate relevant synthetic questions whose content, except for the arguments, might not appear in the text.

Our extraction of comparable relations falls within the field of Relation Extraction, in which CRF is a state-of-the-art method (Mooney and Bunescu, 2005; Culotta et al., 2006). We note that in the works of Jindal and Liu (2006) and Li et. al. (2010) comparative questions are identified as an intermediate step for the task of extracting compared entities, which are unknown in their setting. We, on the other hand, detect the compared entities in a pre-processing step, and our target is the extraction of the comparable relations given known candidate entities.

Our algorithm ranks relevant templates based on the similarity between an article's content and the typical context of each relation. Prior work rank relevant concrete questions to a given input question, focusing on strong lexical similarities (Jeon et al., 2005; Cai et al., 2011; Hao and Agichtein, 2012). We, however, do not expect to find direct lexical similarities between candidate relations and the article. Instead, we are interested in a higher level topical similarity to the input article, for which LDA topics were shown to help (Celikyilmaz et al., 2010).

Finally, several works present unsupervised methods for ranking proper template instantia-

tions, mainly as selectional preferences (Light and Greiff, 2002; Erk, 2007; Ritter et al., 2010). However, we eventually choose instantiation candidates, and thus preferred supervised methods that enable filtering and not just ranking. Furthermore, we target a more subtle discrimination between entities than prior work, *e.g.* between quarterbacks, singers and actors. Machine learning naturally incorporates the many features that capture different aspects of entity characterization.

7 Conclusions

We introduced the novel task of automatically generating synthetic comparable questions that are relevant to a given news article but do not necessarily appear in it. To this end, we proposed an algorithm that consists of two parts. The offline part identifies comparable relations in a large collection of questions. Its output is a database of comparable relations together with a context profile for each relation and models that detect correct instantiations of this relation, all learned from the question corpus. In the online part, given a news article, the algorithm identifies relevant comparable relations based on the similarity between the article content and each relation's context profile. Then, relevant relations are instantiated only with pairs of named entities from the article whose comparison makes sense by applying the instantiation correctness models to candidate pairs.

We assessed the performance of our algorithm via a Mechanical Turk experiment. A partial version of our algorithm, without instantiation filtering, was our strongest baseline. The full algorithm outperformed this baseline by 45% on question correctness, but surprisingly also by 46% on question relevance. These results show that our supervised filtering methods are successful in keeping only correct pairs, but they also serve as an additional filtering for relevant relations, on top of context matching.

In future work, we want to generate more diverse and intriguing questions by selecting relevant named entities for template instantiation that do not appear in the article. Another direction would be take a supervised approach, training classifiers over a labeled dataset for filtering irrelevant templates and incorrect instantiations. Finally, it would be interesting to see how our algorithm performs on other news domains.

References

Manish Agarwal, Rakshit Shah, and Prashanth Mannem. 2011. Automatic question generation using discourse cues. In *Proceedings of the 6th Workshop on Innovative Use of NLP for Building Educational Applications*, IUNLPBEA '11, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, March.

Jonathan C. Brown, Gwen A. Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, pages 819–826, Stroudsburg, PA, USA. Association for Computational Linguistics.

Li Cai, Guangyou Zhou, Kang Liu, and Jun Zhao. 2011. Learning the latent topics for question retrieval in community qa. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 273–281, Chiang Mai, Thailand, November. Asian Federation of Natural Language Processing.

Asli Celikyilmaz, Dilek Hakkani-Tur, and Gokhan Tur. 2010. Lda based similarity modeling for question answering. In *Proceedings of the NAACL HLT 2010 Workshop on Semantic Search*, SS '10, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM TIST*, 2(3):27.

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, HLT-NAACL '06, pages 296–303, Stroudsburg, PA, USA. Association for Computational Linguistics.

Katrin Erk. 2007. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 216–223, Prague, Czech Republic, June. Association for Computational Linguistics.

Tianyong Hao and Eugene Agichtein. 2012. Finding similar questions in collaborative question answering archives: toward bootstrapping-based equivalent pattern learning. *Inf. Retr.*, 15(3-4):332–353, June.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation.

In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 609–617, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiwoon Jeon, W. Bruce Croft, and Joon Ho Lee. 2005. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 84–90, New York, NY, USA. ACM.

Nitin Jindal and Bing Liu. 2006. Mining comparative sentences and relations. In *proceedings of the 21st national conference on Artificial intelligence - Volume 2*, AAAI'06, pages 1331–1336. AAAI Press.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of ICML*.

Shasha Li, Chin-Yew Lin, Young-In Song, and Zhoujun Li. 2010. Comparable entity mining from comparative questions. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 650–658. Association for Computational Linguistics.

Marc Light and Warren R. Greiff. 2002. Statistical models for the induction and use of selectional preferences. *Cognitive Science*, 26(3):269–281.

Tom M. Mitchell. 1997. *Machine learning*. McGraw Hill series in computer science. McGraw-Hill.

Ruslan Mitkov, Le An Ha, and Nikiforos Karamanis. 2006. A computer-aided environment for generating multiple-choice test items. *Nat. Lang. Eng.*, 12(2):177–194, June.

Raymond J. Mooney and Razvan Bunescu. 2005. Mining knowledge from text using information extraction. *SIGKDD Explor. Newsl.*, 7(1):3–10, June.

Niko Myller. 2007. Automatic generation of prediction questions during program visualization. *Electron. Notes Theor. Comput. Sci.*, 178:43–49, July.

A.M. Olney, A.C. Graesser, and N.K. Person. 2012. Question generation from concept maps. *Dialogue & Discourse*, 3(2):75–99.

D. Pollard. 2001. *A User's Guide to Measure Theoretic Probability*. Cambridge University Press.

F. Provost. 2000. Machine learning from imbalanced data sets 101. *Proceedings of the AAAI-2000 Workshop on Imbalanced Data Sets*.

Alan Ritter, Mausam, and Oren Etzioni. 2010. A latent dirichlet allocation method for selectional preferences. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 424–434, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vasile Rus, Brendan Wyse, Paul Piwek, Mihai C. Lintean, Svetlana Stoyanchev, and Cristian Moldovan. 2010. The first question generation shared task evaluation challenge. In John D. Kelleher, Brian Mac Namee, Ielka van der Sluis, Anja Belz, Albert Gatt, and Alexander Koller, editors, *INLG 2010 - Proceedings of the Sixth International Natural Language Generation Conference, July 7-9, 2010, Trim, Co. Meath, Ireland*. The Association for Computer Linguistics.

Anne Schuth, Maarten Marx, and Maarten de Rijke. 2007. Extracting the discussion structure in comments on news-articles. In *Proceedings of the 9th annual ACM international workshop on Web information and data management*, WIDM '07, pages 97–104, New York, NY, USA. ACM.

Frank Wilcoxon. 1945. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1:80–83.