11-2014

Generative Modeling of Entity Comparisons in Text

Maksim TKACHENKO

Hady Wirawan LAUW
Singapore Management University, hadywlauw@smu.edu.sg

Generative Modeling of Entity Comparisons in Text

Maksim Tkachenko*
Dept. of Analytical Information Systems
Saint Petersburg State University
maksim.tkatchenko@gmail.com

Hady W. Lauw
School of Information Systems
Singapore Management University
hadywlauw@smu.edu.sg

ABSTRACT

Users frequently rely on online reviews for decision making. In addition to allowing users to evaluate the quality of individual products, reviews also support comparison shopping. One key user activity is to compare two (or more) products based on a specific aspect. However, making a comparison across two different reviews, written by different authors, is not always equitable due to the different standards and preferences of individual authors. Therefore, we focus instead on comparative sentences, whereby two products are compared directly by a review author within a single sentence.

We study the problem of comparative relation mining. Given a set of comparative sentences, each relating a pair of entities, our objective is two-fold: to interpret the comparative direction in each sentence, and to determine the relative merits of each entity. This requires mining comparative relations at two levels of resolution: at the sentence level, as well as at the entity level. Our key observation is that there is significant synergy between the two levels. We therefore propose a generative model for comparative text, which jointly models comparative directions at the sentence level, and ranking at the entity level. This model is tested comprehensively on Amazon reviews dataset with good empirical outperformance over the state-of-the-art baselines.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; H.2.8 [**Database Applications**]: Data Mining

Keywords

generative model; comparison mining; comparative sentences

1. INTRODUCTION

With the advent of social information processing, we increasingly turn to the wisdom of the crowd in social media to aid our decision making. One decision aid is the ability to easily compare various alternatives. Few existing commercial systems assist users in making a comparison of entities (products) based on other users' experiences. Comparison shopping sites such as Google Shopping or PriceGrabber compare prices among different sellers for the same product. Others, such as DPReview for digital cameras, compare different cameras[1] based on structured attributes of the cameras, instead of users' perception of various aspects.

Our objective is to mine user-generated content to assist users in making well-informed comparisons of entities. Essentially, we view it as a crowdsourcing way to determine the relative quality of entities, from the users' vantage point. In the following discussion, we use online reviews as a representative example of such a user-generated content. Mining reviews for opinions on various products is an active area of research, including the two distinct, yet complementary areas of opinion mining and comparison mining.

Opinion mining [14] deals with sentences expressing user sentiments about an entity. The following sentence from an Amazon review expresses a positive sentiment on the image quality of the camera CANON EOS 50D: *"Based on my initial experience with this Camera, I must say that 50D produces amazingly sharp pictures in low light conditions, with kids, and a variety of outdoor scenarios like sports and nature."*. By aggregating sentiments across such sentences, we can arrive at a sense of the quality of a product. While opinion mining has its use in revealing the strengths and weaknesses of an entity, there are inherent disadvantages in basing a comparison of two entities on opinion sentences about individual entities. For one, the respective opinions may be expressed by different users, potentially with different standards or purposes, rendering the comparison inequitable.

Therefore, we focus on *comparison mining*, whereby the bases for comparison are comparative sentences about two entities. For example, the following sentence expresses a comparison between CANON EOS 50D vs. CANON EOS 40D in terms of image quality: *"The 50D is sharper than my 40D and the images are not soft."*. In this case, the same user (providing a common benchmark) compares two entities within the same context. Table 1 shows several more examples for three pairs of digital cameras. This focus on comparative sentences distinguishes our work from opinion mining, and opens up the potential of other sentences not currently covered by opinion mining. To maintain focus, here we deal with sentences involving two entities, and would explore sentences involving more entities in a future work.

*Work done while visiting Singapore Management University.

CIKM'14, November 3–7, 2014, Shanghai, China.

ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2662016.

[1] `http://www.dpreview.com/products/compare/cameras`

Entity Pair	Example Comparative Sentences
Canon EOS 50D, Canon EOS 40D	s_1 : I am surprised to see that the images on the 40D are better than the 50D. s_2 : And from the research I did it appears the 50D's images can be sharpened and still have more detail than the 40D. s_3 : The 50D is sharper than my 40D and the images are not soft. s_4 : While I'm not prepared to go into a long detailed comparison, I will say that I find the image quality and the auto-focus accuracy of the 50d to be noticeably better than the 40d.
Canon EOS 40D, Canon Rebel XTi	s_5 : I 'm using the same lenses as I did with the XTi and when I did side by side comparisons with the same settings, the images taken with the 40D have much better detail and sharper contrast, giving the photos better depth . s_6 : Although the XTi is a great camera , which I used to capture some really terrific shots , the quality of shots taken with the 40D is vastly superior.
Canon EOS 50D, Canon Rebel XTi	s_7 : I traded up from the XTi to the 50D - better sensor, higher ISO speeds, doubling of burst speed from XTi's 3 fps to 6 fps.

Table 1: Example Comparative Sentences about Digital Cameras from Amazon.com

Problem. Given a corpus of comparative sentences, relating pairs of entities in a particular domain (e.g., digital cameras) on a specific aspect (e.g., image quality), we seek to derive the comparative relations among the entities, i.e., between any two comparable entities, which one is better in that aspect. The corpus of comparative sentences may be obtained from user-generated content expressing user preferences in social media, such as reviews (see Section 6.1.1).

Naturally, the comparative relations need to be modeled at two levels. First, at the level of each comparative sentence. For example, s_1 in Table 1 may be analyzed to determine that it favors Canon EOS 40D. Second, at the level of entity pairs. The sentence-level relations are aggregated to form the comparative relation, e.g., whether Canon EOS 50D is better than Canon EOS 40D. Both levels are important and useful. The latter provides an aggregate view. The former provides supporting evidence at the sentence level.

Solving this problem is challenging. For one, deciphering comparative relation from text is difficult, due to complex structures affecting the comparison (e.g., which entity is mentioned first, negation such as "not soft"), as well as linguistic forms such synonymy (e.g., "better" and "sharper") and polysemy (e.g., "doubling" may connote positively for image quality, but negatively for price). For another, intrepreting the relation between two entities is fraught with uncertainties due to inconsistencies among different writers.

Approach. Traditionally, these challenges are addressed in a fragmented manner by solving the two levels separately as a pipeline, for instance by first determining the sentence-level comparisons, and then aggregating them into the entity-level comparisons (see Section 5). This fragmentation is unnecessary, and could even be deterimental, when errors from one level propagate unmitigated to the other.

We postulate that an *integrated* approach would work better because of the synergy between sentence-level and entity-level comparisons. Intuitively, if one entity is better than another entity, we would expect that many comparative sentences would compare the former favorably with respect to the latter. Thus, knowing which entity is better helps to determine the comparison in a sentence, and vice versa.

We illustrate this intuition with an example in Figure 1 involving 6 sentences (right) on 5 entities $\{A, B, C, D, E\}$. Supposing the meaning of the first four sentences with the word "better" is known, we can confidently rank some pairs, by drawing a bold edge from the worse entity to the better entity, e.g., from B to A, since "A is better than B".

There are a couple of questions that we seek to answer. One question is which of D or E is better, since there is no clue from the bold edges alone. Another question is the meaning of the last two sentences, since we have not yet understood "superior". Considering these two questions separately does not offer an answer. Considering them jointly allows us to arrive at an answer to both. Since $A \leftarrow B \leftarrow C$, by transitivity, we can infer that $A \leftarrow C$ (dotted means inferred). In turn, $A \leftarrow C$ allows us to interpret the sentence "A is superior to C", i.e., that "superior" implies the first-mentioned entity is better. This then allows us to parse the last sentence to infer that $D \leftarrow E$ (dotted). We thus can recover the correct rank order $A \leftarrow B \leftarrow C \leftarrow D \leftarrow E$.

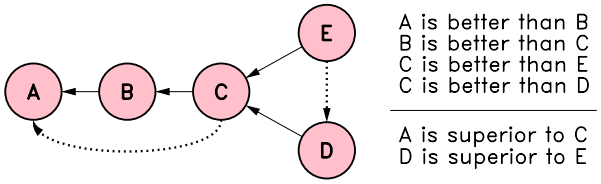

Figure 1: Separate treatment of the tasks does not allow to recover the correct ranking $A \leftarrow B \leftarrow C \leftarrow D \leftarrow E$.

Contributions. We make the following contributions.

First, we propose an integrated approach for comparative relation mining. This is a departure from previous pipelined approaches (see Section 5). It is not necessarily feasible, nor desirable, simply to "stitch up" existing formulations of sentence-level and entity-level comparisons. They are disparate frameworks that are not easy to bring together other than as a disjoint pipeline. We therefore build a novel method from the ground up to solve the problem holistically. Our integrated formulation is presented in Section 2.

Second, we design a generative model (see Section 3), called *CompareGem*, which stands for COMPArative RElation GEnerative Model, and propose an inference algorithm based on Gibbs sampling (see Section 4). We turn to *generative modeling* because it offers several significant advantages. For one, it would be *probabilistic*, suitable for modeling the uncertainties in text. For another, it would be a *joint* model, connecting sentence-level and entity-level comparisons seamlessly. It would also be *generative*, with greater flexibility in accommodating supervised and unsupervised settings.

Third, through experiments on datasets from Amazon reviews (see Section 6), we validate that *CompareGem* indeed outperforms the pipelined baselines in both supervised and unsupervised configurations, underlining the utility of integrated modeling in comparative relation mining.

2. PROBLEM STATEMENT

As input, we consider a set of entities E (e.g., digital cameras). For each pair of entities $e_i, e_j \in E$, S_{ij} denotes the set of comparative sentences involving e_i and e_j (in any order of mention) for a specific aspect (e.g., image quality or price). We can equivalently refer to S_{ij} as S_{ji}. For instance, in Table 1, the pair of entities CANON EOS 50D and CANON EOS 40D are associated with a set of four comparative sentences $\{s_1, s_2, s_3, s_4\}$ on image quality. Some pairs may not have any comparative sentence, if they are never compared by any user, i.e., $S_{ij} = \emptyset$. The union of all comparative sentences is denoted $\mathcal{S} = \bigcup_{e_i, e_j \in E} S_{ij}$. We will describe how $\mathcal{S}$ can be obtained from a corpus of reviews in Section 6.1.1.

Our objective is to learn the comparative relation between any two comparable entities e_i or e_j, in terms of which one is better for the aspect of interest. Using the example of CANON EOS 50D and CANON EOS 40D in Table 1, we see that s_1 favors CANON EOS 40D, whereas s_2 to s_4 favor CANON EOS 50D. The majority consensus favors the conclusion that CANON EOS 50D is better in image quality.

We first discuss the question of when two entities are comparable. In layman's terms, we simply do not wish to compare apples and oranges. We adopt a data-driven approach as follows. If there is at least one comparative sentence in $\mathcal{S}$ about two entities, then some user has deemed them comparable. Comparability is also transitive. If e_i has been compared to e', and e' has been compared to e_j, then we can compare e_i and e_j using e' as a conduit for the comparison.

DEFINITION 1 (COMPARABILITY). *Two entities e_i and e_j are* comparable *if and only if one of the following conditions holds. First, $S_{ij} \neq \emptyset$. Second, $S_{ij} = \emptyset$, but there exist a list of indices $i = k_1, k_2, \ldots, k_{n-1}, k_n = j$ such that every element of the chain $(e_{k_1}, e_{k_2}), (e_{k_2}, e_{k_3}), \ldots, (e_{k_{n-1}}, e_{k_n})$, connecting e_i and e_j, meets the first condition.*

For each comparable pair of entities, we want to determine which one is better. To capture this notion of relative "quality" among entities, we associate each entity e_i with a rank score $a_i \in \mathbb{R}$, where e_i is "better" than e_j if $a_i > a_j$. This rank score is latent, and needs to be learnt from the data. As seen in Table 1, the sentences are not always unanimous in terms of which entity is favored. Even when there is a consensus, there may also be some variance (e.g., dissenting opinions). It is important not just to capture the relation at the entity level, but also the comparative direction at the sentence level to provide a full picture of the comparison.

With the notations in place, we are now ready to state our problem statement formally, as follows.

PROBLEM 1 (COMPARATIVE RELATION MINING). *Given a set of entities E and the associated corpus of comparative sentences $\mathcal{S}$, find:*

- *for every entity $e_i \in E$, its rank score a_i,*
- *for every sentence $s \in \mathcal{S}$ about a comparable pair of entities e_i and e_j, the comparative direction (or comparison outcome) in terms of whether e_i or e_j is favored by s.*

3. MODEL

We first discuss the modeling of comparative sentences, before describing our generative model *CompareGem*.

3.1 Bag of Features

The convention in modeling text, either for classification [26] or topic modeling [1], is to model a document as a bag of words, due to the assumption of exchangeability of words within a document. In other words, only the frequencies of words, and not the sequence in which they occur, matter.

While we also deal with text, the atomic unit of interest is a sentence. A bag-of-words model is not appropriate for modeling a *comparative* sentence. Recognizing the favored entity in a comparative sentence is a challenging problem due to the complex sentence structure, whereby word order now becomes important. Let us consider the following example comparative sentence: *"The 50D is sharper than my 40D"*. In terms of the bag-of-words model, the order between *50D* and *40D* could be swapped interchangeably. However, we know that swapping the order of those two words would change the meaning of the comparison completely.

We observe that, rather than words alone, the clues to the comparison come in the form of features, as follows:

- *Syntactic features.* We distinguish whether a word appears before the first-mentioned entity, in between the two entities, or after the second-mentioned entity. For example, the word "sharper" may translate to a feature $\langle \#1\ sharper\ \#2 \rangle$, where #1 and #2 refer to slots for the first- and second-mentioned entities respectively.
- *Negation features.* It is also important to pay attention to negation. We therefore stick the adverb 'not' to a feature if it occurs close to the target word.

Rather than a bag-of-words model, we model each comparative sentence s as a bag of *features*, where each feature w is drawn from a vocabulary of features W (such as mentioned above). The bag representation also maintains the frequency of each feature within each sentence.

3.2 Generative Model

CompareGem is a generative model for comparative sentences. Its plate notation is shown in Figure 2. First, for each sentence $s \in S$, we generate the comparison outcome (which entity is favored in s). Thereafter, based on the comparison outcome, we generate each feature $w \in s$.

Generating Comparison Outcome. Each sentence s in the corpus $\mathcal{S}$ expresses a comparison outcome involving two entities (say e_i and e_j). We associate each sentence s with a binary random variable c_s. The event $c_s = 0$ is the outcome when the first-mentioned entity is favored, whereas $c_s = 1$ is the outcome when the second-mentioned entity is favored. For simplicity, we do not model a draw, which would not influence the ranking between the two entities.

Since c_s is a random variable, its outcome depends on an underlying probability distribution. As mentioned in Section 2, we associate each entity $e_i \in E$ with a rank score a_i that reflects the quality of e_i. Intuitively, the higher a_i is than a_j, the higher is the probability that a comparative sentence would favor a_i. One suitable way to model this probability for binary outcomes is the sigmoid function, as shown in Equation 1, supposing the first-mentioned entity is e_i and the second-mentioned entity is e_j.

$$\begin{aligned} \mathrm{P}(c_s = 0 | a_i, a_j) &= \mathrm{P}(e_i \text{ is better than } e_j | a_i, a_j) \\ &= \sigma_v(a_i - a_j) = \frac{1}{1 + e^{-v(a_i - a_j)}} \end{aligned} \quad (1)$$

If a_i is significantly higher than a_j, the probability would tend towards 1, reflecting e_i's much higher quality. If $a_i = a_j$, the probability is 0.5, reflecting the uncertain outcome between two evenly matched entities. Conversely, if a_i is significantly lower than a_j, the probability would tend towards 0. The parameter υ models the sensitivity to the difference between the two rank scores. If υ is large, small differences in scores would have a high impact on the probabilities.

There is also the question of the appropriate range of a_i's. One option is to model it along a continuous spectrum, with a Gaussian prior for the distribution of a_i's, which would encode the prior that most entities are probably of "average" rank scores, while some are extremely high or low. However, in some scenarios, the requirement is simply to place each entity along a discrete scale, e.g., 0 to 9, for an easy-to-understand ranking of entities. Another option, which we adopt in this paper, is to have a discretized model, with N number of ranking steps in the scale of 0 to $N-1$. Instead of a Gaussian, the prior can thus be simulated by a binomial distribution $Binomial(N-1, p_0)$, where p_0 is probability of success in a Bernoulli trial ($p_0 = 0.5$ for our model). As $N \to \infty$, we get ever closer to the continuous version. We experiment with different settings of N and υ in Section 6.

Generating Features. Once the comparison outcome c_s for a sentence s is generated, we then generate the features of the sentence. We assume that each outcome $c \in \{0, 1\}$ is associated with a parameter θ_c, which is a probability distribution $\{\mathrm{P}(w|\theta_c)\}_{w \in W}$ over features in the vocabulary W. For instance, θ_0 is a distribution over features for the case when the first-mentioned entity is favored, in which case features involving words such as "better", "sharper" may have higher probabilities. Meanwhile, θ_1 is the same for when the second-mentioned entity is favored, in which case those involving words such as "worse" may be more likely.

Generative Process. We now describe the full generative process of the model.

1. Both θ_0 and θ_1 are sampled from a Dirichlet distribution with α parameter:

$$\theta_0, \theta_1 \sim Dirichlet(\alpha)$$

2. For each entity $e_i \in E$, we sample its rank score a_i:

$$a_i \sim Binomial(N-1, p_0),$$

3. For every comparative sentence $s \in \mathcal{S}$ comparing two entities e_i (first-mentioned) and e_j (second-mentioned):

   (a) Sample the comparison outcome c_s:

$$c_s \sim Bernoulli(\sigma_\upsilon(a_i - a_j)),$$

   (b) Sample each feature w in sentence s:

$$w \sim Categorical(\theta_{c_s})$$

As shown in Figure 2, the only observed (shaded) variables are the features w's within each sentence s. All the other random variables are latent and need to be learnt from the data. The likelihood function of an assignment of scores $A = \{a_i\}_{e_i \in E}$, comparison outcomes $C = \{c_s\}_{s \in \mathcal{S}}$ and latent distributions over features $\theta = \{\theta_0, \theta_1\}$ is shown in Equation 2, where the three main terms correspond to the three main steps in the above generative process.

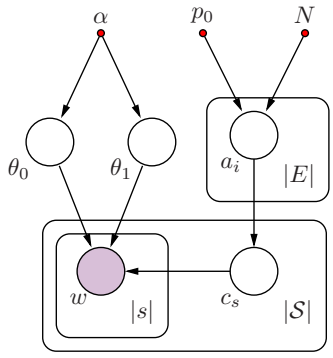

Figure 2: *CompareGem* in Plate Notation.

$$\mathcal{L}(C, A, \theta) = \prod_{c \in \{0,1\}} \mathrm{P}(\theta_c|\alpha) \times \prod_{e_i \in E} \mathrm{P}(a_i|N-1, p_0) \times \prod_{e_i, e_j \in E} \prod_{s \in S_{ij}} \left[\mathrm{P}(c_s|a_i, a_j) \prod_{w \in s} \mathrm{P}(w|\theta_{c_s})\right] \quad (2)$$

Once the model parameters have been learned, we would obtain the solution to the Problem 1 defined in Section 2. The rank scores a_i's would provide the ranking among entities. On the other hand, the comparison outcome of a sentence s can be obtained from the *posterior* distribution of c_s, taking into account the corresponding entities' rank scores a_i and a_j and the features in s, as in Equation 3.

$$\mathrm{P}(c_s|a_i, a_j, s) = \frac{\mathrm{P}(c_s|a_i, a_j) \prod_{w \in s} \mathrm{P}(w|\theta_{c_s})}{\sum_{c' \in \{0,1\}} \mathrm{P}(c'|a_i, a_j) \prod_{w \in s} \mathrm{P}(w|\theta_{c'})} \quad (3)$$

As intuited in Section 1, this joint modeling of entity ranking ($\{a_i\}_{e_i \in E}$) and sentence-level comparison $\{c_s\}_{s \in S}$ is expected to help both tasks. In particular, the effect may be especially significant for the sentence-level comparison, since the ranking may complement the very few features that a sentence usually has. While ranking may be somewhat more robust to sentence-level errors due to aggregating multiple sentences, more accurate sentence-level comparison outcomes are still expected to improve ranking inference.

3.3 Discussions

Unsupervised vs. Supervised. In the above generative process, we have assumed a fully unsupervised setting. We would learn the ranking of entities, and the direction of ranking is interpreted by inspecting the feature distributions. This is because, in theory, the direction of larger score indicating "better" could be swapped (i.e., smaller score coding for "better") while still having the same likelihood.

Our model also accommodates a "supervised" setting. To introduce "light" supervision, we label a subset of sentences in terms of their comparison outcomes. Where in the unsupervised setting, only the w's are observed, in the supervised setting, we would consider some c_s variables (corresponding to a subset of labeled sentences) to also be observed (having known outcomes). This would have the effect of grouping together sentences of the same label, which would then influence the respective feature distributions θ_0 and θ_1. We

will explore both unsupervised and supervised settings in Section 6.

Another possible direction for supervision, which we do not explore in this paper, is to consider the rank score a_i of some entities to be observed. Imposing some ranking order is probably too heavy-handed, because it imposes some form of "opinion" (as opposed to labeling each sentence's comparison outcome, which is more objective). Imposing ranking runs counter to our objective of learning the crowdsourced ranking based on user-generated content.

Relation to Other Learning Models. The modeling of comparison outcomes in Equation 1 has some relation to the family of latent ability models (see Section 5) used to model competitions with known outcomes. In a way, a comparative sentence can be interpreted as a "competition" between two entities, where the outcome is one entity becomes the "winner". The key difference is that in our case the outcomes are latent and unknown, and need to be learned from text. This synergy between competition modeling and generative modeling of text is novel, and we will include a comparison to a baseline of pure competition model in Section 6.

If we were to remove the ranking component altogether, for instance by setting $N = 1$ so that every entity has exactly the same rank score, the model degenerates into a simplified generative model for text clustering or classification, in a way similar to multinomial Naive Bayes model with unbiased probability over favored entities c_s, and with α prior of the words parameters. We therefore include Naive Bayes classification as a baseline for comparison in Section 6.

Our feature distributions are also somewhat related to the notion of "topics" in topic modeling [1], where each topic codes for some concept based on word co-occurences. Several crucial differences set us apart from topic modeling. In our case, there are always two "topics" corresponding to the two comparison outcomes (and not an arbitrary number of topics), the distribution is over features (and not over words), and the primary mechanism for learning is the comparison model in addition to feature co-occurrences (and not word co-occurrences alone as in topic modeling).

4. INFERENCE BY GIBBS SAMPLING

Gibbs sampling [11] provides a mechanism to infer hidden variables of a graphical model. It allows drawing samples from a joint probability distribution of two or more random variables, when direct sampling is intractable. It is a special case of Monte Carlo algorithm that defines a Markov chain in the space of possible variable assignments. We sample one variable at a time from the conditional distribution of that variable, conditioned on all the others. The stationary distribution of the Markov chain is the joint distribution over the variables and samples drawn in a such way are guaranteed from the joint distribution.

For *CompareGem*, we use the collapsed version of Gibbs sampling, by integrating out θ_0, θ_1. The derivation is provided below.

$$\begin{aligned}\mathcal{L}(C, A) &= \int_\theta \mathcal{L}(C, A, \theta) d\theta \\ &= \prod_{e_i \in E} \mathrm{P}(a_i | N-1, p_0) \prod_{e_i, e_j \in E} \prod_{s \in S_{ij}} \mathrm{P}(c_s | a_i, a_j) \times \\ &\int_\theta \prod_{c \in \{0,1\}} \mathrm{P}(\theta_c | \alpha) \prod_{e_i, e_j \in E} \prod_{s \in S_{ij}} \prod_{w \in s} \mathrm{P}(w | \theta_{c_s}) d\theta. \end{aligned} \quad (4)$$

We separately integrate the expression for θ_0 and θ_1. For θ_c, we have:

$$\begin{aligned}\mathrm{P}(\mathcal{S} | \alpha; c) &= \int_{\theta_c} \mathrm{P}(\theta_c | \alpha) \prod_{e_i, e_j \in E} \prod_{s \in S_{ij} | c_s = c} \prod_{w \in s} \mathrm{P}(w | \theta_{c_s}) d\theta_c \\ &= \frac{\Gamma(\alpha K)}{\Gamma^K(\alpha)} \int_{\theta_c} \prod_{k=1}^{K} \theta_c^{\alpha - 1} \prod_{s \in \mathcal{S} | c_s = c} \prod_{w \in s} \prod_{k=1}^{K} \theta_c^{w_k} d\theta_c \\ &= \frac{\Gamma(\alpha K)}{\Gamma^K(\alpha)} \int_{\theta_c} \prod_{k=1}^{K} \theta_c^{n(k,c) + \alpha - 1} d\theta_c \\ &= \frac{\Gamma(\alpha K)}{\Gamma^K(\alpha)} \frac{\prod_{k=1}^{K} \Gamma(n(k,c) + \alpha)}{\Gamma(\alpha K + \sum_{k=1}^{K} n(k,c))}, \end{aligned} \quad (5)$$

where $n(k, c)$ denotes the frequency of the feature w_k within the sentences with comparison outcome $c_s = c$, and K is the size of the feature vocabulary ($K = |W|$).

The other components of the likelihood are shown below.

$$\begin{aligned}&\mathrm{P}(c_s | a_i, a_j) = (1 - \sigma_\upsilon(a_i - a_j))^{c_s} \times \sigma_\upsilon^{1 - c_s}(a_i - a_j) \\ &= \left(1 - (1 + e^{-\upsilon(a_i - a_j)})^{-1}\right)^{c_s} \left(1 + e^{-\upsilon(a_i - a_j)}\right)^{-(1 - c_s)} \\ &= \frac{e^{-c_s \upsilon(a_i - a_j)}}{(1 + e^{-\upsilon(a_i - a_j)})^{c_s}} \times \frac{1}{(1 + e^{-\upsilon(a_i - a_j)})^{1 - c_s}} \\ &= \frac{e^{-c_s \upsilon(a_i - a_j)}}{1 + e^{-\upsilon(a_i - a_j)}}, \end{aligned} \quad (6)$$

$$\mathrm{P}(a_i | N-1, p_0) = \frac{(N-1)!}{a_i!(N - a_i - 1)!} p_0^{a_i} (1 - p_0)^{N - a_i}. \quad (7)$$

The algorithm samples the comparison outcome c_s for each sentence $s \in S$, and the latent scores a_i for each entity $e_i \in E$.

Sampling rank scores A. Fixing the preference assignments for sentences C and assuming $p_0 = 0.5$, we sample latent score a_i for every entity e_i:

$$\begin{aligned}\mathrm{P}(a_i = a | A_{-i}, ...) &\propto \mathrm{P}(a | N-1, p_0 = 0.5) \prod_{e_j \in E} \prod_{s \in S_{ij}} \mathrm{P}(c_s | a_i, a_j) \\ &\propto \frac{1}{a!(N - a - 1)!} \prod_{e_j \in E} \prod_{s \in S_{ij}} \frac{e^{-c_s \upsilon(a - a_j)}}{1 + e^{-\upsilon(a - a_j)}}, \end{aligned} \quad (8)$$

where A_{-i} denotes the set of latent scores for each entity except e_i's.

Sampling comparison outcomes C. Fixing the ranking scores A, we sample the comparison outcome c_s for each sentence s:

$$\begin{aligned}\mathrm{P}(c_s = c | C_{-s}, ...) &\propto \mathrm{P}(c_s | a_i, a_j) \times \prod_{c \in 0,1} \mathrm{P}(\mathcal{S} | \alpha; c) \\ &\propto e^{-c_s \upsilon(a_i - a_j)} \prod_{c \in 0,1} \frac{\prod_{k=1}^{K} \Gamma(n(k,c) + \alpha)}{\Gamma(\alpha K + \sum_{k=1}^{K} n(k,c))} \\ &\propto e^{-c_s \upsilon(a_i - a_j)} \prod_{c \in 0,1} \left[\frac{\prod_{k=1}^{K} \Gamma(\bar{n}(k,c) + \alpha)}{\Gamma(\alpha K + \sum_{k=1}^{K} \bar{n}(k,c))} \times \right. \\ &\left. \frac{\prod_{k=1}^{K} \prod_{l=1}^{f_s(c,k)} (\bar{n}(c,k) + \alpha + l - 1)}{\prod_{l=1}^{f_s(c)} (\alpha K + \sum_{k=1}^{K} \bar{n}(c,k) + l - 1)} \right] \\ &\propto e^{-c_s \upsilon(a_i - a_j)} \prod_{c \in 0,1} \frac{\prod_{k=1}^{K} \prod_{l=1}^{f_s(c,k)} (\bar{n}(c,k) + \alpha + l - 1)}{\prod_{l=1}^{f_s(c,\cdot)} (\alpha K + \sum_{k=1}^{K} \bar{n}(c,k) + l - 1)}, \end{aligned} \quad (9)$$

where $\bar{n}(k, c)$ returns the count of the feature k within all the sentences labeled with c excluding the sampled sentence s; $f_s(c, k)$ denotes the frequency of feature k in the sentence s for the assignment c; $f_s(c, \cdot) = \sum_{k=1}^{K} f_s(c, k)$.

Although Gibbs sampling allows estimating the shape of a probability distribution, one can modify this process to maximize the model likelihood. We used *simulated annealing*, the technique used in optimization to find global optimum of a given (non-convex) function. We sample each variable from the modified distribution:

$$\mathrm{P}(v_j = v|...) \rightarrow \frac{\mathrm{P}(v_j = v|...)^{1/t_j}}{\sum_v \mathrm{P}(v_j = v|...)^{1/t_j}}, \qquad (10)$$

where the sequence $T = (t_j)_{j=1}^{n}$ defines the cooling schedule and a particular value t_j is called the temperature. As $t_j \rightarrow 0$ the distribution becomes sharper (setting $t_j = 1$ for every j recovers standard Gibbs sampling procedure) and the modified distribution concentrates all the mass on the maximal outcome.

Each iteration of the Gibbs sampler takes $\mathcal{O}(|E||\mathcal{S}|)$ time. It is easy to see that at each iteration the algorithm samples comparison outcomes C for the sentence set $\mathcal{S}$, which requires $\mathcal{O}(|\mathcal{S}|)$ operations. The rank score sampling involves $\mathcal{O}(|S_{ij}|)$ time for each entity e_i, which can be bounded by $\mathcal{O}(|\mathcal{S}|)$. Indeed, if the number of sentences for every entity is evenly distributed, the term $|E|$ can be dropped, then the iteration time is linear in the number of sentences $\mathcal{O}(|\mathcal{S}|)$.

5. RELATED WORK

We first review the related work in comparison mining to define our baselines, then compare to other related problems.

Among the earliest task being considered in comparison mining is the identification of comparative sentences from a corpus containing both comparative and non-comparative sentences [17, 9]. Once the comparative sentences have been identified, the next task is to extract the entities being compared within each sentence [18, 19], and to resolve mentions of the same entity across sentences [6]. These tasks are orthogonal, and yet complementary to our problem, and we discuss how we deal with these issues in Section 6.1.1.

Our focus in this work is on *mining comparative relations*. Given a pair of entities, and their relevant comparative sentences, which entity is better? This requires an examination of comparative relation at two levels: sentences and entity pairs. As mentioned in Section 1, these two levels have traditionally been studied separately, as follows.

At the sentence level, the objective is to determine which of the two entities being mentioned is considered better. Previous work either uses training labels in supervised classification [33], or known indicators such as "pros" and "cons" within reviews [10]. Since supervised classification is the more recent and more general approach, here we consider two classifiers as baselines: Support Vector Machine (SVM) [5] and Naive Bayes (NB) [26], as implemented in Weka [12], using the same features as described in Section 3.

At the entity pair level, the objective is to determine which of the two entities is better overall [21, 36, 23]. This is done by aggregating the sentence-level comparisons into an overall ranking of entities. The main approach in previous work is to build a graph of entities, with a directed edge from one entity to another entity, weighted by the number of sentences that claims the latter is better than the former [21]. The ranking is then derived using a network centrality measure such as PageRank (PR) [27]. We use this as the first ranking baseline, implemented according to [21].

For the second ranking baseline, we consider latent ability models, used to aggregate pairwise comparisons in education [30], sports [7], and gaming [13]. The input is a set of pairwise comparisons or matches with known binary outcomes (which entity wins), and the output is the rank score for each entity. As baseline, we will use the Bradley-Terry-Luce model (BTL) [3, 25], which shares a similar sigmoid-based probability of winning as our model. This method has not been employed for the task of mining comparisons from text, but we include it as a pseudo-baseline for completeness.

Different from comparison mining, opinion mining focuses on opinions or sentiments on *individual* products [14]. It is frequently decomposed into two sub-tasks, namely: identifying the aspect or feature being described [15], as well as determining the sentiment expressed [24]. These are generally modeled as classification problems, but there are also unsupervised variants based on topic modeling [28, 4]. Other than sentiments, there are also works focusing on modeling the correlation betweeen topics and user ratings [32].

Comparative summarization addresses the problem of summarizing two (or more) *separate* corpora in terms of a comparison. This is a different setting from ours, where each pair of entities are compared within a sentence. One formulation of comparative summarization is *sentence alignment*, which selects pairs of sentences (one from each corpus), so that each pair of sentences describes the same "aspect" [31]. In some cases, it is desired that sentences within a pair are contrastive [20, 29]. Another formulation of comparative summarization is *comparative topic modeling*, where the objective is to identify different "viewpoints" of a topic [35].

Competitor mining deals with identifying the set of competitors of a given entity, commonly formulated as finding relationships [16] or similarities among entities [34, 22].

6. EXPERIMENTS

Our objective is to study the effectiveness of *CompareGem* on two tasks: comparative direction classification, and entity ranking. Our focus here is on effectiveness, rather than efficiency. All experiments complete within three minutes on a PC with Intel Core i5 CPU 3.2 GHz and 4GB RAM.

6.1 Experimental Setup

6.1.1 Dataset

The corpus of comparative sentences $\mathcal{S}$ can be obtained from text corpora that contain user evaluation of pairs of products, such as online reviews. We crawled reviews from the Digital Cameras category of *Amazon*. Not all sentences within the reviews are comparative. It is not our objective to develop a new method to process reviews into comparative sentences. Below, we describe a methodology that we have used for extracting comparative sentences from reviews, which is followed by manual inspection to ensure a high quality of the dataset. There are three key information that we need to determine: whether a sentence is comparative, the entities being compared, and the aspect of interest.

Entity Recognition. Finding the mentions of objects of a particular type (e.g. cameras, laptops) in text is called named entity recognition (NER). There is no ready-made NER system for the domain we are considering (digital cam-

Aspect	# sentences	#1 is favored (%)	#2 is favored (%)
Functionality	457	38.5	61.5
Form Factor	78	61.3	38.7
Image Quality	129	58.1	41.9
Price	165	52.1	47.9

Table 2: Dataset Size involving 180 Digital Cameras

eras). Therefore, we employ a dictionary matching approach for entity recognition that we find works well in tying the mentions of an object together. We construct the dictionary of entities from product titles, which we employ to perform token-based partial matching search. We then train a decision tree classifier to filter out false positives.

Comparative Sentence & Aspect Identification. Our scope covers sentences that contain two product mentions. We are only concerned with sentences that express a preference for the first mentioned product or for the second one. In addition, we pick the four most frequent aspects mentioned in the reviews, namely: *functionality*, *form factor*, *image quality*, and *price*. To identify whether a sentence is a comparative, and the aspect of interest, we employ a supervised classification approach. To train the respective SVM classifiers for comparative sentence identification and aspect identification, we take 1000 randomly selected sentences, and manually annotate them. We apply this classifier to the rest of sentences with two product mentions. To reduce false positives, the classifier labels are followed by manual inspection.

Aspect identification could be done as preprocessing step with the use of LDA [1] or any other suitable methods. Instead, we manually annotated the sentences to guarantee a high quality of annotations and coherence with the specification benchmark (to be introduced later).

Table 2 shows the dataset sizes. In total, the number of products being compared within extracted sentences is 180. The four aspects respectively have 457, 78, 129, and 165 comparative sentences. Each aspect is a distinct instance of the problem. The distributions between the two classes (whether the first-mentioned (#1) or second-mentioned (#2) entity is favored) are relatively well-balanced. These data sizes are significant, in light of the need to carefully annotate the data, not just with labels, but also with ranking benchmarks (see Section 6.1.3). This is much larger than that used in the previous work on entity ranking [21].

6.1.2 Evaluation Tasks and Metrics

We evaluate the performance of *CompareGem* on two tasks. Every evaluation is conducted in both supervised and unsupervised configurations. We begin with the supervised one.

Comparative Direction Classification. In the first task, all the competing algorithms are given a set of labeled (training) and a set of unlabeled (test) data. Each algorithm is required to identify the favored entity for each comparative sentence in the test data. One can look at this essentially as a binary classification problem. To measure the performance of an algorithm, we calculate its *classification accuracy*, i.e., the fraction of correctly classified sentences (over the total number of sentences in the test set).

Entity Ranking. In the second task, we want to assess the quality of ranking scores produced by the competing algorithms. It is not always feasible to have a ground truth in the form of a ranking list, because some pairs may not be comparable, or there may not be sufficient evidence for some pairs. Therefore, we assume that the ground truth (see Section 6.1.3) has the form of a set of entity pairs X, where the favored (higher-ranked) entity for each pair in X is known. We thus transform the ranking scores output by each algorithm into a set of ordered pairs Y, which we then compare in terms of its agreement with the ground truth X.

As evaluation metric, we express the *ranking accuracy* as the agreement between the ground truth X and the output Y in terms of the fraction of concordant pairs over all pairs in the intersection, expressed as a percentage. Given two sets of ordered entity pairs X and Y, where the first element in a pair is favored over the second, two pairs $(a, b) \in X$ and $(a', b') \in Y$ are concordant if $a = a'$ and $b = b'$.

Ranking accuracy is related to Kendall's tau [8], which can be defined as the number of exchanges needed in a bubble sort to convert one rank to the other. Kendall's tau is defined for the totally ordered sets, and, in this case, its normalized value equals the inverse $(1-)$ ranking accuracy. Unlike Kendall's tau, the proposed metric accepts partially ordered sets, and, thus more suitable here, as comparison makes sense only for comparable entities (see Definition 1).

Unsupervised Configuration. In the unsupervised configuration, no labeled data is used as input for the model. Therefore, the first task resembles clustering into two clusters, rather than classification. We can still use the labels to evaluate this clustering, by computing *purity* instead. Each cluster is "classified" to the majority comparison direction label among sentences in that cluster. We then determine the "classification accuracy" as before. Since our model also outputs the ranking scores, we simply determine the ranking accuracy with respect to benchmarks as before. For unsupervised, whether higher or lower ranking score represents "better" is not known in advance. We check the ranking accuracy in both directions, and take the maximum value.

6.1.3 Entity Ranking Benchmarks

Because there is no single definitive ranking ground truth, we introduce two ranking benchmarks that together provide a more complete picture of our performance in ranking. The first is *specification benchmark*, which is based on very objective hard numerical attributes. The second is *crowdsourced benchmark*, which is based on the subjective opinions of many users (expressed in the comparative sentences).

Specification Benchmark. The intuition is that users' preferences can be traced to some specific attribute of the entities. We collect product specification information from dpreview.com[2] and wikipedia.com. For *form factor*, we say that entity e_i is better than e_j if both the volume and weight of e_i are smaller than the volume and weight of e_j. For *functionality*, we consider the entity with the later release date to be better, assuming that the newer model is more functional than the older one (comparison is done only within product lines, e.g., EOS 50D and EOS 60D). To ensure that the functionality of products has indeed changed, we only consider differences of more than one year. For *price*, we consider the entity with the lower price to be better. To be conservative against price fluctuations, we only consider differences of more than 1000USD. The specification benchmarks contain 291 entity pairs for *functionality*, 5836 pairs for *form factor*, and 1479 pairs for *price*.

[2] *Digital Photography Review* has a large database with detailed information about individual digital cameras.

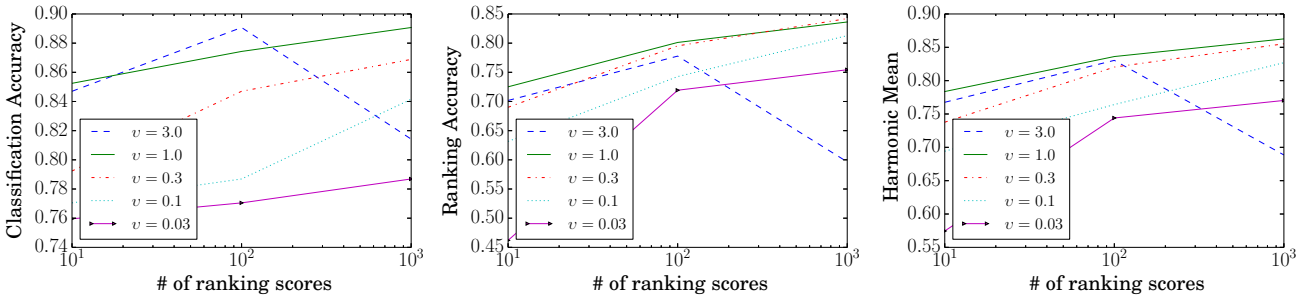

Figure 3: Parameter analysis for *CompareGem*. Horizontal axes represent N. Every plot represents a υ.

It is not our intention to claim that specification benchmark alone is sufficient validation, which is why we consider a second unrelated benchmark below. Rather, we use specification benchmark to show that *CompareGem* is able to recall objective information from texts written by users. In addition, this specification benchmark is independent of the data. Doing well on this benchmark provides some confidence that the method may do well for other aspects without suitable specification benchmark, e.g., *image quality*.

Crowdsourced Benchmark. This benchmark is created from the complete set of labels used for comparative direction classification (the first task). For each pair of entities, we consider each comparative sentence to vote based on its label. The entity with the majority votes is considered better. Therefore, this benchmark reflects how users in general rank these entities, which may not necessarily always be consistent with the specifications. The crowdsourced benchmarks contain 175 entity pairs for *functionality*, 53 pairs for *form factor*, 102 pairs for *price*, and 90 pairs for *image quality*. It is smaller than specification benchmark since it is defined only for pairs that have been compared in the data.

6.2 Parameter Analysis

CompareGem has two parameters: 1) the number of ranking scores N and 2) the ranking scale υ (see Section 3). To select the appropriate parameter settings, we conduct a grid search across various settings of $N \in \{10, 100, 1000\}$ and $\upsilon \in \{0.03, 0.1, 0.3, 1.0, 3.0\}$. We run parameter selection with 50:50 random split into training and test data. We observe similar results for various aspects, and will show the results for *functionality* as a representative.

Figure 3(a) shows classification accuracies. As we increase N, the general trend is that of increasing classification accuracy. With more ranking scores, the ranking is more refined, which may help the accuracy of classification. The exception is only when υ is too high, e.g., $\upsilon = 3$, as the sigmoid function approaches 1 or 0 very rapidly even with small differences between rank scores. This washes out the effects of latent rank scores, because the prior binomial distribution starts to play a big role and tends to concentrate ranking scores close to its mode. Meanwhile, if υ is too low, the performance is also worse, because the model becomes insensitive to small changes in rank scores.

Similar trends, with similar observations, also apply to the ranking accuracies with respect to the specification benchmark shown in Figure 3(b). To get parameter settings that balance out both classification accuracy and ranking accuracy, we combine the two measures with harmonic mean formula ($H(x, y) = \frac{2xy}{x+y}$), shown in Figure 3(c). Evidently, we get the best performance for the setting of $N = 1000$ and $\upsilon = 1.0$, which we will use in the following experiments.

6.3 Comparison to Baselines

6.3.1 *Supervised Configuration*

The aim is to understand how well *CompareGem* tackles the classification and ranking tasks in the presence of training data. We repeat every experiment 10 times on different data shuffles (cross-validation), and average all the accuracy values. The training and test data split is 50:50.

Baselines. As discussed in Section 5, for the classification task, we compare to two popular classification models: Naive Bayes (*NB*) and Support Vector Machine (*SVM*). For the ranking task, our baselines are PageRank (*PR*) and *BTL*. Because *PR* and *BTL* assume the comparison outcomes of sentences are known, we use the classification output from the first task, together with the training sentences as inputs to *PR* and *BTL* respectively. For this reason, neither PageRank and BTL is a complete baseline, because they cannot operate independently from a source of comparative directions. Therefore, for ranking, we create four composite baselines from pipelining the two separate steps discussed in this section, namely: *SVM+PR*, *NB+PR*, *SVM+BTL*, and *NB+BTL*. In contrast, since *CompareGem* is a generative model, we simply learn the two tasks simultaneously.

For all experiments, we conduct randomization test [2] at 5% statistical significance level for the differences between methods. The best result among methods is in bold. Lower results with statistically insignificant differences are shown in italics. Regular font indicates significantly worse results.

Aspect	CompareGem	SVM	NB
Functionality	**89.0**	76.6	74.4
Form Factor	**71.5**	57.8	62.8
Image Quality	**73.8**	65.4	64.5
Price	**68.7**	52.8	55.2

Table 3: Supervised: Classification Accuracy

Classification Accuracy. For the classification task, we report the accuracies of all three methods in Table 3. The clear observation is that *CompareGem* performs significantly better than both *SVM* and *NB* for all aspects. This validates our hypothesis that jointly modeling ranking and classification helps the model do better at classifying sentences. While *NB* and *SVM* classify only one test example

Aspect	Compare Gem	SVM+BTL	NB+BTL	SVM+PR	NB+PR
Functionality	**89.7**	88.6	*88.8*	84.1	84.1
Form Factor	**82.7**	*79.8*	**82.7**	78.2	80.2
Image Quality	**80.7**	78.7	*80.6*	75.9	76.9
Price	**79.0**	75.8	*76.7*	70.6	72.3

Table 4: Supervised: Ranking (Crowdsourced)

Aspect	Compare Gem	SVM+BTL	NB+BTL	SVM+PR	NB+PR
Functionality	*84.0*	**87.0**	*86.0*	75.3	76.0
Form Factor	**68.3**	*68.1*	*67.8*	*66.1*	*65.7*
Price	**82.7**	77.8	77.1	70.6	70.8

Table 5: Supervised: Ranking (Specification)

at a time, *CompareGem* takes advantage of jointly modeling test examples, and finds label assignments maximizing the a posteriori probability of the entire test collection.

Ranking Accuracy. Table 4 shows the ranking accuracies for the crowdsourced benchmark. *CompareGem* has the highest ranking accuracies overall. It is better than the *+PR* models, which have appeared in previous literature, and the difference is statistically significant. We have also introduced *+BTL* models as pseudo-baselines, though they have not appeared in previous literature. *CompareGem* still outperforms *SVM+BTL* significantly in most aspects. With respect to *NB+BTL*, *CompareGem* is a shade better, but not significantly so. We hypothesize that ranking is an "easier" task than classification. Though *SVM* and *NB* perform significantly worse in classification at the sentence level (Table 3), at the level of entity pairs, there could be sufficient number of correctly classified sentences to get the ranking.

Table 5 shows the ranking accuracies for the specification benchmark. Against this benchmark, *CompareGem* still performs well for *form factor* and *price*. For *functionality*, it is slightly worse than *SVM+BTL*, but not statistically significantly so. Between *SVM+BTL* and *NB+BTL*, we now see that the former performs slightly better, which is an opposite trend to the crowdsourced benchmark. This is because the two benchmarks are made from two independent sources of information. There are cases when they disagree. For example, based on the specification benchmark, the newer model of Canon Powershot G10 is better than the older model Canon Powershot G2. However, based on the crowdsourced benchmark, customers actually favor the older model Canon Powershot G2. Overall, the two benchmarks are quite consistent, as shown by *CompareGem*'s high ranking accuracies for both benchmarks.

6.3.2 Unsupervised Configuration

Baselines. In the unsupervised configuration, we do not use any labeled data to classify sentences and induce product ranking. As baseline, we will use *K-means* clustering ($K = 2$) to perform the first task. For ranking, we determine the comparative direction of each cluster based on the majority label, which we put into *PR* and *BTL*, creating two composite baselines: *K-means+BTL* and *K-means+PR*.

Purity. Table 6 shows purity or "classification accuracy". Again, it shows that *CompareGem* is significantly better than the baseline *K-means*. To give further insight, we also report results for majority baseline (*Majority*) whereby all test examples go into the most frequent class. Interestingly, *K-means* does not always outperform *Majority*.

Aspect	CompareGem	K-means	Majority
Functionality	**70.1**	58.9	61.5
Form Factor	**64.9**	59.1	61.3
Image Quality	**64.3**	57.8	58.1
Price	**55.9**	52.1	52.1

Table 6: Unsupervised: Purity

Aspect	CompareGem	K-means+BTL	K-means+PR
Functionality	**65.6**	*64.0*	57.0
Form Factor	**62.9**	*59.8*	*59.4*
Image Quality	**59.4**	53.3	54.0
Price	*55.0*	*54.6*	**56.2**

Table 7: Unsupervised: Ranking (Crowdsourced)

Ranking Accuracy. Table 7 and Table 8 show the ranking accuracies for the crowdsourced and specification benchmarks respectively. In both tables, *CompareGem* tends to have the highest accuracies, except for *price* in Table 7 (where *CompareGem* is the second, but not significantly worse). *CompareGem*'s outperformance is statistically significant in the specification benchmark for all aspects, and in the crowdsourced benchmark for *image quality* aspect.

Comparing the results of supervised vs. unsupervised configurations, we see that the unsupervised results are indeed lower, as expected. Interestingly, the absolute classification and ranking accuracy values are still relatively good (∼60%), which is still a reasonable performance in the case where training labels are unavailable or very difficult to obtain.

6.4 Feature Analysis

To gain further insight into the workings of *CompareGem*, here we investigate the features that play an important role in the supervised model. Since there are two binary classes ($c = 0$ indicating the first-mentioned entity #1 in a sentence is favored, as well as $c = 1$ indicating the second-mentioned entity #2 is favored), we focus on features that are most discriminative between the two classes. A *discriminative* feature w is one whose conditional probability $\mathrm{P}(c|w) \geq 0.8$.

In Table 9, we show the top five features most frequent among sentences assigned to each class, for various aspects. For each feature, #1 and #2 refer to the relative positions of the first- and second-mentioned entities, with respect to a word. For *functionality*, the top feature for $c = 0$, is "#1 from #2", while that for $c = 1$ is "from #1 #2". These involve the same word "from", with different relative positions with respect to the entities. This underlines the importance of the bag-of-features model (see Section 3), as words alone are probably uninformative as features ("from" appears in both classes). A similar case exists for *image quality*, with "#1 better #2" (for $c = 0$) vs. "#1 #2 better" (for $c = 1$).

Other than their relative positions, the actual words that help make up a feature also matter. Interestingly, for *form factor*, we see contrasting features such as "#1 lighter #2" (for $c = 0$) vs. "#1 heavier #2" (for $c = 1$). For *price*, we see "#1 less #2" (for $c = 0$) vs. "#1 more #2" (for $c = 1$).

Aspect	CompareGem	K-means+BTL	K-means+PR
Functionality	**76.7**	67.8	58.4
Form Factor	**62.4**	57.3	51.9
Price	**64.3**	57.3	54.4

Table 8: Unsupervised: Ranking (Specification)

Functionality		Form Factor		Image Quality		Price	
#1 is favored ($c = 0$)	#2 is favored ($c = 1$)	#1 is favored ($c = 0$)	#2 is favored ($c = 1$)	#1 is favored ($c = 0$)	#2 is favored ($c = 1$)	#1 is favored ($c = 0$)	#2 is favored ($c = 1$)
#1 from #2	from #1 #2	#1 lighter #2	#1 heavier #2	#1 than #2	#1 #2 better	#1 less #2	#1 more #2
#1 than #2	#1 #2 upgrad	#1 smaller #2	#1 #2 hand	#1 better #2	#1 us #2	purchas #1 #2	#1 cost #2
#1 over #2	had #1 #2	size #1 #2	#1 larger #2	qualiti #1 #2	#1 #2 detail	#1 better #2	about #1 #2
#1 better #2	#1 decid #2	#1 had #2	#1 #2 feel	pictur #1 #2	#1 #2 which	#1 instead #2	would #1 #2
#1 ha #2	#1 year #2	#1 over #2	#1 #2 lighter	imag #1 #2	#1 out #2	x #1 #2	camera #1 #2

Table 9: Top 5 Most Frequent Discriminative Features

7. CONCLUSION

We study the problem of comparative relation mining, and propose *CompareGem* as a generative model for comparative sentences. The key insight is jointly modeling two levels of comparative relations: at the level of individual sentences as well as at the level of entity pairs. This holistic treatment of comparative relation mining is novel, and is shown to empirically outperform the previous pipelined approaches.

CompareGem is validated comprehensively on Amazon reviews dataset. Comparison to baselines in both supervised and unsupervised configurations show that *CompareGem* is especially effective for the comparative direction task at the sentence level, outperforming all the baselines significantly and decisively. For the entity ranking task, *CompareGem* still produces the highest ranking accuracies, but in some cases the differences to the baselines are relatively close.

This result is revelatory, suggesting that while joint modeling of entity ranking and sentence classification is useful for both tasks, the extents of the benefits are asymmetric. Entity ranking helps sentence classification more than the reverse. Nevertheless, these experiments still convincingly show the utility of *CompareGem* in terms of the two tasks, as well as the flexibility of *CompareGem* in dealing with both supervised and unsupervised configurations.

8. REFERENCES

[1] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *JMLR*, 2003.

[2] G. E. Box, J. S. Hunter, and W. G. Hunter. *Statistics for experimenters*. Wiley New York, 2005.

[3] R. A. Bradley and M. E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 1952.

[4] S. Brody and N. Elhadad. An unsupervised aspect-sentiment model for online reviews. In *NAACL HLT*, 2010.

[5] C. Cortes and V. Vapnik. Support vector machine. *Machine Learning*, 1995.

[6] X. Ding, B. Liu, and L. Zhang. Entity discovery and assignment for opinion mining applications. In *KDD*, 2009.

[7] A. E. Elo. *The rating of chessplayers, past and present*, volume 3. Batsford London, 1978.

[8] R. Fagin, R. Kumar, and D. Sivakumar. Comparing top K lists. In *SODA*, 2003.

[9] R. Feldman, M. Fresko, J. Goldenberg, O. Netzer, and L. Ungar. Extracting product comparisons from discussion boards. In *ICDM*, 2007.

[10] M. Ganapathibhotla and B. Liu. Mining opinions in comparative sentences. In *COLING*, 2008.

[11] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the bayesian restoration of images. *TPAMI*, 1984.

[12] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 2009.

[13] R. Herbrich, T. Minka, and T. Graepel. TrueSkill: A bayesian skill rating system. In *NIPS*, 2006.

[14] M. Hu and B. Liu. Mining and summarizing customer reviews. In *KDD*, 2004.

[15] M. Hu and B. Liu. Mining opinion features in customer reviews. In *AAAI*, 2004.

[16] M. Jang, J.-w. Park, and S.-w. Hwang. Predictive mining of comparable entities from the web. In *AAAI*, 2012.

[17] N. Jindal and B. Liu. Identifying comparative sentences in text documents. In *SIGIR*, 2006.

[18] N. Jindal and B. Liu. Mining comparative sentences and relations. In *AAAI*, 2006.

[19] W. Kessler and J. Kuhn. Detection of product comparisons - how far does an out-of-the-box semantic role labeling system take you? In *EMNLP*, 2013.

[20] H. D. Kim and C. Zhai. Generating comparative summaries of contradictory opinions in text. In *CIKM*, 2009.

[21] T. Kurashima, K. Bessho, H. Toda, T. Uchiyama, and R. Kataoka. Ranking entities using comparative relations. In *DEXA*, 2008.

[22] T. Lappas, G. Valkanas, and D. Gunopulos. Efficient and domain-invariant competitor mining. In *KDD*, 2012.

[23] S. Li, Z.-J. Zha, Z. Ming, M. Wang, T.-S. Chua, J. Guo, and W. Xu. Product comparison using comparative relations. In *SIGIR*, 2011.

[24] B. Liu and L. Zhang. A survey of opinion mining and sentiment analysis. In *Mining Text Data*. Springer, 2012.

[25] R. D. Luce. *Individual choice behavior: A theoretical analysis*. Courier Dover Publications, 2012.

[26] A. McCallum, K. Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[27] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[28] M. Paul and R. Girju. A two-dimensional topic-aspect model for discovering multi-faceted topics. In *AAAI*, 2010.

[29] M. J. Paul, C. Zhai, and R. Girju. Summarizing contrastive viewpoints in opinionated text. In *EMNLP*, 2010.

[30] G. Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. University of Chicago Press, 1981.

[31] R. Sipos and T. Joachims. Generating comparative summaries from reviews. In *CIKM*, 2013.

[32] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, 2011.

[33] K. Xu, S. S. Liao, R. Y. Lau, H. Tang, and S. Wang. Building comparative product relation maps by mining consumer opinions on the web. In *AMCIS*, 2009.

[34] Y. Yang, J. Tang, J. Keomany, Y. Zhao, J. Li, Y. Ding, T. Li, and L. Wang. Mining competitive relationships by learning across heterogeneous networks. In *CIKM*, 2012.

[35] C. Zhai, A. Velivelli, and B. Yu. A cross-collection mixture model for comparative text mining. In *KDD*, 2004.

[36] Z. Zhang, C. Guo, and P. Goes. Product comparison networks for competitive analysis of online word-of-mouth. *TMIS*, 3(4), 2013.