Building a Lightweight Semantic Model for Unsupervised Information Extraction on Short Listings

Doo Soon Kim
Accenture Technology Lab
50 W San Fernando St.,
San Jose, CA, 95113
{doo.soon.kim, k.verma, peter.z.yeh}@accenture.com

Kunal Verma
Accenture Technology Lab
50 W San Fernando St.,
San Jose, CA, 95113

Peter Z. Yeh
Accenture Technology Lab
50 W San Fernando St.,
San Jose, CA, 95113

Abstract

Short listings such as classified ads or product listings abound on the web. If a computer can reliably extract information from them, it will greatly benefit a variety of applications. Short listings are, however, challenging to process due to their informal styles. In this paper, we present an unsupervised information extraction system for short listings. Given a corpus of listings, the system builds a semantic model that represents typical objects and their attributes in the domain of the corpus, and then uses the model to extract information. Two key features in the system are a semantic parser that extracts objects and their attributes and a listing-focused clustering module that helps group together extracted tokens of same type. Our evaluation shows that the semantic model learned by these two modules is effective across multiple domains.

1 Introduction

Short listings such as classified ads or product listings are prevalent on the web. These texts are generally concise – around 10 words in length. Fig. 1 shows some example listings. Due to the recent explosive growth of such listings, extracting information from them becomes crucial for tasks such as faceted search and reasoning. For example, consider an online shopping site on which information about merchandises for sale is posted. Detecting brands/styles/features that are frequently mentioned in the postings would allow a company to design a better marketing strategy.

Most Information Extraction (IE) techniques developed for formal texts, however, would be inapplicable to listings because of their informal and idiosyncratic styles. For example, typos, abbreviations and synonyms often appear and should be resolved (e.g., apartment/apt, bike/bicycle). Symbols could have the special meanings (e.g., x in 2x2 in Fig. 1 indicates number of bedrooms and bathrooms). Tokenization based only on space is insufficient (e.g., RawlingBaseball in Fig. 1). Multiwords such as granite top should also be detected. Applying off-the-shelf parsers is infeasible because of unusual phrasal forms in most listings, such as a long sequence of nouns/adjectives (e.g., “New Paint Wood Floors New Windows Gated Complex”).

To address these challenges, several approaches have applied machine learning algorithms (Ghani et al., 2006) (Putthividhya and Hu, 2011) or an external knowledge base (Michelson and Knoblock, 2005). These approaches, however, commonly require human supervision to produce training data or to build a knowledge base. This is expensive, requiring repeated manual effort whenever a new domain or a new set of information to be extracted is introduced.

In this paper, we present an unsupervised IE system for listings. The system extracts tokens from a corpus of listings and then clusters tokens of the same types, where each resulting cluster corresponds to an information type (e.g., size, brand, etc.). For formal texts, contexts (e.g., surrounding words) have been a major feature for word clustering (Turney and Pantel, 2010). This feature alone, however, is insufficient for short listings because of lack of contextual clues in short listings.
To address this limitation of context-based clustering, we first identify common types of information (main objects and their attributes) represented in listings and apply customized clustering for these types. Specifically, we define a semantic model to explicitly represent these information types, and based on the semantic model, develop two components to improve clustering – a shallow semantic parser and listing-focused clustering module. The semantic parser specifically focuses on extracting main objects and the listing-focused clustering module helps group together extracted tokens of the same type.

Our evaluation shows that our two main contributions (shallow semantic parser and listing-focused clustering) significantly improve performance across three different domains – Craigslist, EBay, and FatWallet. Our system achieves .50~.65 F1-score for the EBay and the FatWallet datasets based on gold standards constructed by human annotators. For the Craigslist dataset, which is more difficult than the other two, F1-score is .35.

2 Related Work

**IE on Listings.** (Ghani et al., 2006) (Putthividhya and Hu, 2011) propose semi-supervised approaches to extract product types and their attributes from product listings. (Ghani et al., 2006) applies the EM (Expectation-Maximization) algorithm to incorporate unlabelled data. (Putthividhya and Hu, 2011) uses unlabelled data to build dictionaries of values to be extracted (e.g., brand or model names), which are then used as a feature for a machine learning system. (Michelson and Knoblock, 2005) uses a manually-crafted knowledge base called reference set to define standard forms of values to be extracted. Using a string edit function, the system then identifies tokens in listings that have a low distance score with the values defined in the reference set. They also propose a semi-supervised method to building reference sets (Michelson and Knoblock, 2009). Unlike these systems, our approach is unsupervised.

**Unsupervised Information Extraction.** Most unsupervised IE systems produce clusters for tokens of same type extracted from a corpus of unlabelled texts. (Chambers and Jurafsky, 2011) (Poon and Domingos, 2010) (Chen et al., 2011) focus on extracting frame-like structures (Baker et al., 1998) by defining two types of clusters, event clusters and role clusters. Event clusters define an event (a situation or a frame) such as BOMBING by clustering verbs/nominalized verbs such as \{kill, explosion\}. Role clusters define the semantic roles of the event (e.g., \{terrorist, gunman\} for the Perpetrator role in BOMBING). Similarly, our system defines two types of clusters – the main concept clusters (e.g., TV or book) and the attribute clusters (e.g., size, color). (Chambers and Jurafsky, 2011) is similar to our approach in that it learns a semantic model, called template, from unlabelled news articles and then uses the template to extract information.

Our system is different because it focuses on informal listings, which the components (such as a parser) used by these systems cannot handle.

**Field Segmentation (Sequence Modelling).** This task focuses on segmenting a short text, such as bibliographies or listings. (Grenager et al., 2005) presents an unsupervised HMM based on the observation that the segmented fields tend to be of multiple words length. (Haghighi and Klein, 2006) exploits prototype words (e.g., close, near, shopping for the NEIGHBORHOOD attribute) in an unsupervised setting. (Chang et al., 2007) incorporates domain specific constraints in semi-supervised learning. Our task is different than these systems because we focus on extracting information to enable a variety of automated applications such as business
intelligence reporting, faceted search or automated reasoning, rather than segmenting the text. The segmented fields are often a long unstructured text (e.g., 2 bath 1 bed for size rather than 2 for BathroomNum and 1 for BedroomNum).

**IE on Informal Texts.** IE on informal texts is getting much attention because of the recent explosive growth of these texts. The informal texts that are attempted for IE include online forums (Gruhl et al., 2009), SMS (Beaufort et al., 2010), twitter messages (Liu et al., 2011).

### 3 Our Approach

Given a corpus of listings for a domain of interest, our system constructs a semantic model that represents the types of information and their values to be extracted. Our system then uses the resulting model to extract both the type and value from the corpus. We first describe the semantic model and then the two key steps for creating this model – shallow semantic parsing and listing-focused clustering. Fig. 3 illustrates these two steps with example listings.

#### 3.1 Semantic Model

Our semantic model captures two important pieces of information – the main concept and its attributes. This representation is based on the observation that most listings (e.g., rentals, products) describe the attributes of a single object (i.e., the main concept in our model). Our system takes advantage of this observation by applying customized clustering for each type of information in the model, which results in better performance compared to a one-size-fits-all algorithm. Moreover, this model is general enough to be applicable across a wide range of domains. We quantitatively show both benefits in our evaluation.

Fig. 2 illustrates our model along with an instantiation for rental listings. + indicates multiple clusters can be created.

![Semantic Model and Instantiation](image)

Figure 2: Semantic model and its instantiation for rental listings. + indicates multiple clusters can be created.

#### 3.2 Shallow Semantic Parser

Our Shallow Semantic Parser (SSP) analyzes an input corpus to produce a partial semantic model. SSP first performs preprocessing and multiword detection. SSP then identifies which resulting tokens are the main concepts and which are their attributes. Qualitative attributes capture numeric values (e.g., 1 bedroom, 150 Hz, and 70 kg), and are generally a number followed by a token indicating the attribute (e.g., unit of measurement). Hence, clusters for qualitative attributes include these indicator tokens (see Fig. 2).

Qualitative attributes capture descriptions about the main concept (e.g., address, shipping information, condition). The values of these attributes generally appear in listings without explicitly mentioning the names of these attributes. Hence, the clusters for qualitative attributes include tokens corresponding to the values themselves (e.g., washer hookup).

#### 3.2.1 Preprocessing and Multiword Detection

SSP preprocesses the corpus through three steps. (1) SSP cleans the corpus by removing duplicate listings and HTML expressions/tags. (2) SSP tokenizes each listing based on spaces along with custom heuristics – e.g., handling alpha-numeric tokens starting with numbers (e.g., 3bedroom to 3 bedroom) and mixed case tokens (e.g., NikeShoes to Nike Shoes). (3) SSP performs POS tagging using an off-the-shelf tagger (Tsuruoka and Tsujii, 2005). To improve accuracy, SSP assigns to a token the most frequent POS across all occurrences of that token. This heuristic works well because most tokens in focused domains, like listings, have only one POS.

SSP then detects multiword tokens based on the following rules:

---

1To handle string variations (e.g., typos) during extraction, our system uses a string edit distance function, Jaro-Winkler distance (Winkler, 1990) with a threshold, 0.9.
1. If a bigram (e.g., top floor) in a listing frequently appears as either a single or dashed token (e.g., TopFloor or top-floor) in other listings, then the bigram is regarded as a multiword.

2. For each bigram, \( w_1 \) \( w_2 \) (excluding symbols and numbers), if the conditional probability of the bigram given either \( w_1 \) or \( w_2 \) (i.e., \( p(w_1|w_2) \) ) is high (over 0.75 in our system), the bigram is considered as a candidate multiword. This rule tests the tendency of two tokens appearing together when either one appears.

   However, this test alone is insufficient, as it often generates coarse-grained results – e.g., baseball glove, softball glove, and Hi-Def TV. To prevent this problem, for each \( w_2 \), we measure the entropy over the distribution of the tokens in the \( w_1 \) position. Our intuition is that high variability in the \( w_1 \) position (i.e., high entropy) indicates that the multiword is likely a breakable phrase. Hence, those candidates with high entropy are removed.

   SPP repeatedly applies the above rules to acquire multiwords of arbitrary length. In our implementation, we limit multiword detection up to four-gram.

3.2.2 Main Concept Identification

SPP then identifies the main concepts (\( mc\_\text{words} \)) and their attributes (\( \text{attrs} \)) to produce a partial semantic model. This process is guided by the observation that main concepts tend to appear as head nouns in a listing and attributes as the modifiers of these head nouns (see the examples in Fig. 3).

Algorithm 1 describes the discovery process of \( mc\_\text{words} \) and \( \text{attrs} \). First, SPP initializes \( \text{attrs} \) with tokens that are likely to be a modifier (line 2), by choosing tokens that frequently appear as the object of a preposition within the corpus – e.g., for rent, with washer and dryer, for baseball.

SPP then iteratively performs two steps – \( \text{PARSE} \) and \( \text{EXPANDMODEL} \) (lines 3 ~ 6) – in a bootstrap manner (see Fig. 4). \( \text{PARSE} \) tags the noun tokens in each listing as either head nouns or modifiers. Specifically, \( \text{PARSE} \) first assesses if a listing is “hard” to parse (line 10) based on two criteria – (1) the listing contains a long sequence of nouns (seven words or longer in our system) without any prepositions (e.g., worth shutout series 12” womans fast-pitch softball fielders glove s0120 lefty); and (2) the majority of these nouns do not appear in \( mc\_\text{words} \) and \( \text{attrs} \) (e.g., over 70% in our system). The listings meeting these criteria are generally difficult to recognize the head noun without any semantic knowledge. \( \text{PARSE} \) will revisit these listings in the next round as more \( mc\_\text{words} \) and \( \text{attrs} \) are identified.

If a listing does not meet these criteria, \( \text{PARSE} \) tags nouns appearing in \( mc\_\text{words} \) and \( \text{attrs} \) as head nouns and modifiers respectively (line 11). If this step fails to recognize a head noun, a heuristic is used to identify the head noun – it identifies the first noun phrase by finding a sequence of nouns/adjectives/numbers, and then tags as the head noun the last noun in the phrase that is not tagged as a modifier (line 13). For example, in the first listing of Fig. 3, brentwood apts. is the first noun phrase that meets the condition above; and hence apt. is tagged as the head noun. The remaining untagged nouns in the listing are tagged as modifiers (line 15).
Algorithm 1 Extracting main concepts

1: Input: POS-tagged corpus, corp
2: Initialize attrs
3: repeat
4: \((hn,mod) = \text{Parse}(corp, mc\_words, attrs)\)
5: \((mc\_words,attrs) = \text{ExpandModel}(hn,mod)\)
6: until mc\_words, attrs not changed
7:
8: function Parse(mc\_words, attrs)
9: for all each listing do
10: if parsable then
11: Parse with mc\_words, attrs
12: if headnoun is not tagged then
13: Tag the last noun in the first noun phrase that are not a modifier as hn
14: end if
15: Tag the other nouns as mod
16: end if
17: end for
18: end function
19:
20: function ExpandModel(corp)
21: For each token, calculate a ratio of as a head noun to as a modifier
22: Add tokens with high ratio to mc\_words
23: Add tokens with low ratio to attrs
24: end function

ExpandModel assigns tokens to either mc\_words or attrs based on the tags generated by Parse. For each token, ExpandModel counts the frequency of the token being tagged as a head noun and as a modifier. If a token is predominately tagged as a head noun (or a modifier), the token is added to mc\_words (or attrs) \(^3\).

This bootstrap method is advantageous \(^4\) because SSP can initially focus on easy cases – i.e., mc\_words and attrs that can be detected with high confidence, such as condo(mc\_words) and bedroom(attrs), which often appear as a head noun and a modifier in the easy-to-parse listings. These results can help the system to parse more difficult listings. For example, identifying condo(mc\_words) helps parsing the following more difficult listing, “2 bedroom 1 bathroom condo large patio washer dryer available” – condo would be tagged as a head noun and the rest of the nouns as modifiers.

The result of this step is a partial semantic model that contains a cluster for the main concept and a list of candidate attribute tokens.

3.3 Listing-Focused Clustering

Listing-Focused Clustering (LFC) further expands the partial semantic model (constructed by SSP) by grouping the remaining candidate attribute tokens into attribute clusters – i.e. one cluster for each attribute of the main concept. LFC may also add a token to the main concept cluster if appropriate.

For formal texts, distributional similarity is widely used for clustering words because the contextual clues in these texts are sufficiently discriminative (Lin and Pantel, 2001). This feature alone, however, is insufficient for listings because they lack discriminative contexts due to the short length. Hence, our approach augments context-based similarity with the following rules (presented in order of precedence), based on general properties we observed from listing data across various domains:

- Two quantitative attribute tokens cannot be placed into the same cluster if they frequently appear together in a listing. For example, bed and bath should not be clustered because they frequently appear together (e.g. 2 bed / 2 bath). This rule is based on the observation that a quantitative attribute is likely to appear only once in a listing. To enforce this restriction, for all pairs of tokens, \(t_1\) and \(t_2\), LFC measures the conditional probability of the pair appearing together in a listing given the appearance of either

\(^3\)In our system, if the ratio of the frequency of the head noun to the frequency of the modifier is over .55, the token is added to mc\_words. If less than .35, it is added to attrs.

\(^4\)The bootstrapping cycle generally ends within 3~4 iterations.
$t_1$ and $t_2$. If any of these conditional probabilities are high, $t_1$ and $t_2$ are not clustered.

- Attribute types are strongly enforced by never clustering together a quantitative attribute token and a qualitative attribute token. The type of a token is determined by analyzing the immediate preceding tokens throughout the corpus. If the preceding tokens are generally numbers, then LFC regards the token as a quantitative attribute. Otherwise, the token is regarded as a qualitative attribute.

- Two tokens are similar if the characters in one token appear in the other, preserving the order (e.g., `bdrm` and `bedroom`)

If the above rules fail to determine the similarity between two tokens, LFC reverts to context-based similarity. For each token, LFC creates a context vector containing frequencies of the context words around the token with a window size of two. For example, in the first sentence in Fig. 3, the context words around `apts.` are `l-start` (beginning of the sentence), `l-brentwood`, `r-with`, and `r-3`\(^6\). The frequencies in these vectors are also weighted using PMI scores (pointwise mutual information) between a token and its context words, as suggested by (Turney and Pantel, 2010). The intuition is that a high PMI indicates a context word is strongly associated with a token and hence has high discriminative power. We also apply a smoothing function suggested in (Turney and Pantel, 2010) to mitigate PMI’s bias towards infrequent events. The similarity score is based on a cosine similarity between the two weighted vectors.

Based on this similarity function, LFC applies agglomerative clustering (with average linkage) to produce attribute clusters (or to expand the main concept cluster). However, calculating similarity scores for all pairs of tokens is expensive. To address this problem, LFC performs clustering in two steps. First, LFC performs agglomerative clustering on all pairs of tokens with high frequency\(^6\). LFC then calculates the similarity between each low-frequency token and the clusters resulting from the previous step. If the similarity score is over a user-specified threshold, then LFC adds the token to the cluster. If the score is less than the threshold but the token still appears relatively frequently, then LFC creates a new cluster for the token.

### 4 Evaluation

We perform two evaluations to assess the performance of our approach. First, we evaluate how well our approach extracts the correct type (e.g., `BedroomNum`) and value (e.g., 2) across multiple domains. Next, we evaluate the contribution of each main component in our approach – i.e., shallow semantic parsing and listing-focused clustering – through an ablation study. We also provide an in-depth error analysis along with how our system’s performance is affected by the corpus size.

#### 4.1 Evaluation Setup

We first assemble three different listing datasets for our evaluation – housing rental advertisements from Craigslist (`Rent Ad`), auction listings of baseball gloves from eBay (`Glove`), and hot deal postings of TV/Projector from FatWallet (`TV Deal`)\(^7\). Table 1 shows the size of each dataset, and example listings are shown in Fig. 1\(^8\). We also include example extractions for each domain in Table 3.

We then construct a gold standard by employing two independent human annotators. To do this, we first define the information types (i.e. main concept and attribute) for each dataset (and hence the targets for extraction). We use attributes from eBay and eBay.

---

\(^5\)and $r$ indicates the left or right window.

\(^6\)The threshold for stopping clustering is determined with the development dataset.

\(^7\)The datasets are available at https://sites.google.com/site/2soonk/

\(^8\)The parameters were tuned by the development set. Our system is sensitive to the similarity score threshold in agglomerative clustering but less sensitive to the other parameters. Hence, we tuned the similarity threshold for each domain while fixing the values for the other parameters across different domains.

### Table 1: Dev/Test indicates the number of listings used for development/testing. Avg word indicates the average number of words in a listing. The development dataset is used to tune the parameters.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dev</th>
<th>Test</th>
<th>Avg word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rent Ad</td>
<td>8,950</td>
<td>9,400</td>
<td>9.44</td>
</tr>
<tr>
<td>Glove</td>
<td>8,600</td>
<td>9,500</td>
<td>10.56</td>
</tr>
<tr>
<td>TV Deal</td>
<td>-</td>
<td>900</td>
<td>15.60</td>
</tr>
</tbody>
</table>

---

1086
as the starting point for Glove and TV Deal; and attributes from Rent.com for Rent Ad. We review these attributes with two independent Subject Matter Experts (SMEs) to identify (and include) additional, missing attributes that are useful for analytics and reporting. In total, 19 types (Rent Ad), 15 types (Glove) and 19 types (TV Deal) are defined. For each dataset, we randomly select 100 listings from the Test listings, and instruct each annotator to extract values from these listings, based on the information types for the dataset. Table 2 shows the defined attributes of each data set and the inter-annotator agreement across the attributes.

Finally, we apply our system to the Test listings in each dataset; and evaluate the extraction results against the gold standards using the metrics of precision (P), recall (R), and F1-score (a harmonic mean of precision and recall). Each extraction result is a tuple \((T_i, v_i)\) where \(T_i\) is the information type (e.g., BedroomNum) and \(v_i\) is the value (e.g., 2).

We say that an extraction result is correct if \(v_i\) matches exactly the value extracted by the annotator and \(T_i\) matches the information type assigned to \(v_i\) by the annotator. To enforce this criteria, we match the attribute clusters produced by our system to the information types. This matching step is needed because our approach is unsupervised and hence the clusters are unlabelled. We use two methods for this matching step – many-to-one mapping and one-to-one mapping. For many-to-one mapping (as used in (Chen et al., 2011)), we match an attribute cluster to the information type whose values (as extracted by the annotator) have the highest overlap with the

---

### Table 2: Information types for each domain. See Fig. 1 for the examples of each type. Due to space limitation, we show only top five types in terms of the number of extractions made by the annotators. The parentheses indicate inter-annotator agreement based on exact match between two annotators (first number) and partial match (second number).

<table>
<thead>
<tr>
<th>Domain</th>
<th>Housing Type</th>
<th>Location</th>
<th>Neighborhood</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rent Ad</td>
<td>housing_type</td>
<td>studio, townhome, condo, townhouse, cottage, bd, bed, br, bedroom, bdrm, bedrooms</td>
<td>downtown, San Francisco, china town</td>
<td></td>
</tr>
<tr>
<td>Glove</td>
<td>product_type</td>
<td>mitt, base mitt, glove, glove mitt</td>
<td>rawlings, louisville slugger, mizuno, wilson</td>
<td></td>
</tr>
<tr>
<td>TV Deal</td>
<td>product_type</td>
<td>hdtv, wall mount, monitor</td>
<td>wallmart, amazon, newegg, best buy</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: Examples of positive extractions for the main concept attributes (e.g., housing_type, product_type), the qualitative attributes (e.g., num_bedroom, size) and the quantitative attributes (e.g., neighborhood, brand, seller) used in the experiment.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Housing Type</th>
<th>Location</th>
<th>Neighborhood</th>
<th>Others</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rent Ad</td>
<td>num_bedroom</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Glove</td>
<td>size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TV Deal</td>
<td>seller</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

9http://www.rent.com

10We use Cohen’s kappa, \(\frac{P(a)-P(e)}{1-P(e)}\), where \(P(a)\) is the agreement probability, is calculated by the number of listings in which the two annotators agree divided by 100. \(P(e)\), the chance agreement probability, is calculated by \(\sum_{(T_i, v_i)} P_e((T_i, v_i)) \times P_e((T_i, v_i))\) in which \(P_e((T_i, v_i))\) denotes a probability of extracting \(v_i\) of the type \(T_i\) by the annotator \(j\). \(P((T_i, v_i))\) is calculated by the frequency of \((T_i, v_i)\) extracted divided by the frequency of \(T_i\) extracted.
values of the cluster. Hence, many attribute clusters can map to the same information type. This method, however, has one major disadvantage: high performance can be easily achieved by creating small-sized clusters – e.g., singleton clusters in the extreme case. To mitigate this problem, we also use a one-to-one mapping method – i.e. at most one attribute cluster (i.e. the one with the best overlap) can be mapped to one information type.

We report results for both methods. We also report results for partial matches using the same metrics above. We say that an extraction result is partially correct if \( v_i \) partially matches the value extracted by the annotator (hardwood vs. hardwood floors) and \( T_i \) matches the information type assigned to \( v_i \) by the annotator.\(^{11}\)

### 4.2 Performance Across Multiple Domains

Table 4 shows the system performance result across the domains. Considering our approach is unsupervised, the result is encouraging. For the baseball glove and the TV dataset, F-score is .51 and .66 (in one-to-one mapping). For the rent ad, which is more difficult, F-score is .39. We hypothesize that the low F1-score in the rent ad dataset is the result of poor extraction due to spurious tokens (e.g., our community, this weather). To test this hypothesis, we measure the performance of our system only on the extraction task (i.e., excluding the information type assignment task). Table 5 shows that the performance on extraction for the rent ad dataset is the lowest, confirming our hypothesis.

We also measure the performance per each information type. Fig. 5 shows the result, revealing several facts. First, the main concept clusters (housing_type and product_type) achieve a high F1-score, showing the benefit of our semantic parser. Second, the quantitative attributes (e.g., num_bedroom, glove_size, screen_size) generally have a higher F1 than the qualitative attributes (e.g., location, neighborhood, series_name). These qualitative attributes, in fact, have a low inter-annotator agreement (e.g., .64, .55 for location and neighborhood in Rent Ad and .51 for series_name in Glove), indicating the difficulty of exactly predicting the extractions made by the annotators. If we consider the partial match or the extraction only match (Ext) for those qualitative attributes, their F1-scores are significantly higher than the exact match in the Full task.

### 4.3 Ablation Study

To evaluate our semantic parser and listing-focused clustering module, we ablate these two components to create four versions of our system for comparison – Baseline, Baseline+LFC, Baseline+SSP and Full System. Baseline performs a space-based tokenization followed by clustering based only on the context feature. Baseline+LFC and Baseline+SSP add listing-focused clustering and shallow semantic parser features respectively. The Full system uses both features.

\(^{11}\)We could not compare our system to other systems for several reasons. First, to the best of our knowledge, no unsupervised IE system has been built specifically for short listings. Second, semi-supervised systems such as (Putthividhya and Hu, 2011) (Michelson and Knoblock, 2005) require domain-specific dictionaries, which are expensive to build and scale. Third, even developing a supervised IE system is non-trivial. In our preliminary evaluation with the (linear chain) conditional random field (170/30 training/testing listings) using basic features (the lexemes and POS of the current word and words in the two left/right windows), precision/recall/F1-score are .5/.33/.4. This result is no better than our system. More training data and/or better features seem to be required.
Figure 5: F1-score across the attribute types shown in Table 2. The parentheses indicate the number of extractions made by our system and the annotators respectively.

Table 6: Based on one-to-one mapping. * indicates two-tail statistically significant difference (p < 0.05) against Baseline in Fisher’s test. † indicates one-tail difference. The Fisher’s test is inapplicable to F1-scores.

Table 7: The top five errors based on the one-to-one mapping.
The results shown in Table 6 lead to the following observations. First, the use of SSP (Baseline+SSP) makes an improvement for all categories except Full(Par) in the glove dataset. This is because SSP identifies main concepts accurately. Second, while LFC by itself (Baseline+LFC) is effective only in the glove dataset, it has the best F1-score in all three domains when combined with SSP (Full System). The result also shows that the simple space-based tokenization and the context-based clustering (Baseline) is insufficient for handling short listings.

4.4 Error Analysis and Corpus Size Effect

We analyze the error types for the wrong extraction made by our system. Specifically, for each error, we assign it to one (or more) of the following causes: (1) a cluster (and hence attribute type) was excluded due to the 1-to-1 mapping methodology described above (NOT IN MAPPING); (2) the value extracted by the system was not extracted by any of the annotators (WRONG EXT); (3) wrong information type – i.e., the token belonged to a wrong cluster (WRONG TYPE); (4) incorrect tokenization for an information type (TK-<type name>).

Table 7 shows the result. In all three domains, NOT IN MAPPING is a major source of error, indicating the system’s clusters are too fine-grained as compared to the gold standard. WRONG EXT is another source of error (especially in the housing rental), indicating the system should extract more informative tokens. Tokenization on the qualitative attributes (neighborhood, series name, display technology in Table 7) should be improved also.

Finally, we measure the effect of the corpus size on the system performance. Fig. 6 shows how the F1-score varies with the corpus size 12. It shows that a small corpus size is sufficient for achieving good performance. We hypothesize that, for focused domains such as our dataset, only a couple of hundred listings are sufficient to acquire meaningful statistics.

5 Conclusion and Future Work

We presented an unsupervised IE system on short listings. The key features in our system are a shallow semantic parser and a listing-focused clustering module. Our evaluation shows the benefits of the two features across multiple domains. To improve our system further, we plan the following works. First, we plan to compare our system with supervised systems to identify the gap between the two systems. Second, as in (Poon and Domingos, 2010), we plan to explore a joint learning method to combine the tasks of tokenization, forming the main concept cluster and forming the attribute clusters; these tasks depend on the outputs of one another. Finally, we plan to explore that external knowledge resources such as DBPedia (Auer et al., 2007) and FreeBase (Bollacker et al., 2008) can be used to further improve performance.

6 Acknowledgements

We would like to thank Colin Puri and Rey Vasquez for their contribution to this work. We also thank the anonymous reviewers for their helpful comments and suggestions for improving the paper.

References


ACL ’98, pages 86–90, Stroudsburg, PA, USA. Association for Computational Linguistics.
Dekang Lin and Patrick Pantel. 2001. Dirt@sbt@discovery of inference rules from text. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, KDD ’01, pages 323–328, New York, NY, USA. ACM.


