Structured Aspect Extraction

Omer Gunes[†]
[†] Department of Computer Science
University of Oxford
United Kingdom
first.last@cs.ox.ac.uk

Tim Furche[†‡]
[‡] Meltwater
Shoreditch, London
United Kingdom
first.last@meltwater.com

Giorgio Orsi[*‡]
[*]School of Computer Science
University of Birmingham
United Kingdom
g.orsi@cs.bham.ac.uk

Abstract

Aspect extraction identifies relevant features of an entity from a textual description and is typically targeted to product reviews, and other types of short text, as an enabling task for, e.g., opinion mining and information retrieval. Current aspect extraction methods mostly focus on aspect terms, often neglecting associated modifiers or embedding them in the aspect terms without proper distinction. Moreover, flat syntactic structures are often assumed, resulting in inaccurate extractions of complex aspects. This paper studies the problem of structured aspect extraction, a variant of traditional aspect extraction aiming at a fine-grained extraction of complex (i.e., hierarchical) aspects. We propose an unsupervised and scalable method for structured aspect extraction consisting of statistical noun phrase clustering, cPMI-based noun phrase segmentation, and hierarchical pattern induction. Our evaluation shows a substantial improvement over existing methods in terms of both quality and computational efficiency.

1 Introduction

The abundance of web data has made information extraction (IE) of strategic importance for data-driven companies such as, e.g., retailers, because of its applications to information retrieval (Kannan et al., 2011), knowledge base construction (Shin et al., 2015), media intelligence (Zeng et al., 2010), and conversational agents (Cassell, 2000). Among all IE-related tasks, *Aspect extraction* (AE) (Zhang and Liu, 2014) is certainly one of the most challenging. AE aims at identifying features of an entity, e.g., a phone, from a free text description ans is commonly associated with (aspect-based) sentiment analysis as a means of extracting *aspect terms* and associated *opinions* in, e.g., product reviews or other forms of opinionated and evaluative text (Hu and Liu, 2004; Popescu and Etzioni, 2005). Similar methods have also been used for extracting *attributes* from, e.g., product listings and other forms of *descriptive* text (Ghani et al., 2006; Kannan et al., 2011). An obvious challenge in aspect extraction is dealing with noisy, unstructured text (NUT). NUT has consistently challenged traditional NLP tools, in particular POS taggers and dependency parsers, due to ungrammatical sentences and incorrect orthography.

Example 1: Aspect extraction

Victorian two bedroom mid terrace property located in Cambridge and comprising of living room with ORIGINAL!!! cupboards, and ORIGINAL!!! picture rail.Stairway off living room leads to two bedrooms.

↓

{ bedroom mid terrace, picture rail.Stairway, cupboards, bedrooms, property} {Cambridge, ORIGINAL}

Example 1 shows the output of IBM's Alchemy Language service[1] when asked to retrieve aspect terms and named entities. Alchemy Language extracts picture rail.Stairway as an aspect due to a missing space after the full stop, and ORIGINAL as a named entity, due to incorrect orthography. Another challenge is distinguishing aspect terms from their *modifiers*. The terms Victorian, mid terrace, and the (nested) expression two bedroom are *qualifying* the term property, while the term two *quantifies* the term bedroom.

[1]https://alchemy-language-demo.mybluemix.net as of July 2016.

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 2321–2332, Osaka, Japan, December 11-17 2016.

Problem definition This paper studies the problem of recognizing and extracting *structured* aspects. *Structured Aspect Extraction* (SAE) generalizes aspect term extraction (ATE) with the extraction and *linking* of modifiers of the aspect term, identifying the possibly hierarchical structure of the aspects. Consider the text Victorian two bedroom mid terrace property from Example 1. An SAE system is expected to produce a single extraction with explicit annotations for *(i)* the aspect terms, i.e., property, bedroom *(ii)* the aspect modifiers, correctly typed as qualifiers, i.e., Victorian, mid terrace, and quantifiers, i.e., two, and *(iii)* the hierarchical structure of the aspect, i.e., $\langle\{\text{Victorian}, \langle\{\text{two}\}, \text{bedroom}\rangle, \text{mid terrace}\}, \text{property}\rangle$.

Contributions (Zhang and Liu, 2014) termed the recognition of *flat* $\langle\text{modifier}, \text{aspect}\rangle$ structures (i.e., a simple form of SAE) a very challenging task. In this paper we go beyond that, proposing (Section 2) a method for unsupervised SAE consisting of: **(1)** An effective normalization strategy specifically engineered for NUT. **(2)** A statistical noun phrase clustering method for unsupervised discovery of aspect terms and (raw) modifiers. **(3)** A cPMI-based noun-phrase segmentation, to identify multi-word expressions and nested structures. **(4)** A controlled induction of *structured aspect patterns*. We evaluate our method against state-of-the-art aspect extraction systems using data from the widely-accepted SemEval benchmarks and a SAE-specific dataset. The evaluation (Section 3) shows that our method improves on existing unsupervised systems for aspect-term extraction and solves the harder SAE task reliably.

Positioning Existing work on aspect extraction focuses mostly on three tasks: **(1)** Entity extraction (Zhang and Liu, 2014; Yahya et al., 2014), i.e., classifying entities in a free-text fragment (e.g., car, phone). **(2)** Aspect term extraction (Yu et al., 2011; Zhang and Liu, 2014; Chen et al., 2014), i.e., extracting features of the entity (e.g., battery, performance). **(3)** Opinion extraction (Pang and Lee, 2008; Yang and Cardie, 2014), i.e., extracting sentiments or opinions and link them to aspects terms.

(Probst et al., 2007) was the first method to go beyond simple aspect terms, proposing a semi-supervised method for extracting $\langle\text{modifier}, \text{aspect}\rangle$ pairs from product descriptions. Despite being supervised, the method has issues distinguishing genuine pairs representing aspects from spurious ones. (Kelly et al., 2012) makes extractions more precise by targeting $\langle\text{modifier}, \text{relation}, \text{aspect}\rangle$ triples in verbal phrases. Although precise, applying this method is unrealistic in practice since it severely impairs recall. In fact, empirical evidence suggests that most of the interesting aspects occur in proximity of named entities and in noun phrases. This shows the need for more flexibility in extracting aspects.

Leveraging a dependency parser is a possible way to capture more complex aspect structures, or at least to provide clues about the presence of entity modifiers (e.g., via AMOD relations). However, deep parsing is known to be inaccurate on NUT, and training suitable models difficult, since NUT lacks proper grammatical structure in the first place. This has been clear since the works by (Popescu and Etzioni, 2005) and (Raju et al., 2009) proposing, as a solution, ATE methods based on noun-phrase clustering. More recently, (Kim et al., 2012) used noun-phrase clustering to extract $\langle\text{modifier}, \text{aspect}\rangle$ pairs where modifiers are arbitrary n-grams, generically typed as either modifiers or quantifiers. Although limited to flat structures, this is currently the closest work to SAE we are aware of.

Similarly to (Popescu and Etzioni, 2005; Raju et al., 2009), we use *frequency*-based noun-phrase clustering to detect aspect terms in an unlabelled corpus. Besides making clustering more accurate by normalizing the descriptions, our method detects the underlying structure of the aspect via a cPMI-based segmentation of noun phrases that simulates an aspect-oriented parsing of the sentence. PMI values are also used by (Popescu and Etzioni, 2005) but only for aspect-term detection. Another option is to use topic modeling for clustering (Titov and McDonald, 2008; Sauper and Barzilay, 2013; Zhang and Liu, 2014). Topic modeling is effective in ATE but can under-cluster when applied to SAE, e.g., noun phrases headed by bedroom and kitchen are clustered together despite having different modifiers, producing incorrect associations of modifiers to aspect terms.

The ability to identify arbitrary hierarchical structures distinguishes our work from other aspect extraction methods based on distributional analysis of syntactic features such as, e.g., (Qiu et al., 2011; Zhuang et al., 2006; Wu et al., 2009; Yu et al., 2011; Zhou et al., 2013; Liu et al., 2013), and rule-based methods such as, e.g., (Poria et al., 2014). Our method can deal with a wide range of aspect structures without resorting to supervision as commonly done in methods based on sequence labelling, such as, e.g., (Li et

al., 2010; Choi and Cardie, 2010; Jakob and Gurevych, 2010; Yang and Cardie, 2014). Our hierarchical extraction should not be confused with the mapping of flat aspects to taxonomies and knowledge bases as done, e.g., by (Yahya et al., 2014). The generalization of these hierarchical structures into patterns enables us to transfer these structures to the extracted aspects, thus labelling the extractions as in (Kim et al., 2012), distinguishing the aspect term, from its qualifiers and quantifiers. Let us stress out that these coarse hierarchical structures cannot be trivially derived from fine-grained dependency relations.

2 Structured Aspect Extraction

We now formally define the task of *structured aspect extraction* (SAE). We introduce some non-standard notation but we assume readers are familiar with aspect extraction terminology.

Definitions A *structured aspect* is a tuple $\langle \mathsf{M}, \mathsf{t} \rangle$, where t is an *aspect term* (or *head* of the aspect) and $\mathsf{M}=\{\mathsf{m}_1, \ldots, \mathsf{m}_n\}$ is a sequence of modifiers of the aspect term (or *tail* of the aspect). A labelling function $\lambda : \mathsf{M} \cup \{\mathsf{t}\} \mapsto \mathsf{L}$ maps aspect terms and modifiers to a set L of labels. Aspect terms are labelled as the real-world entities they represent, e.g., BEDROOM. Each modifier $\mathsf{m} \in \mathsf{M}$ is labelled as either $\lambda(\mathsf{t})$-QUANTIFIER or $\lambda(\mathsf{t})$-QUALIFIER where $\lambda(\mathsf{t})$ is the label of the aspect term t. Consider the text fragment Victorian two bedroom mid terrace property from Example 1. The corresponding SAE is $\langle \{\text{Victorian}, \langle \{\text{two}\}, \text{bedroom} \rangle, \text{mid terrace}\}, \text{property} \rangle$ where the aspect terms property and bedroom are labelled as PROPERTY and BEDROOM respectively, Victorian, mid terrace, and $\langle \{\text{two}\}, \text{bedroom} \rangle$ are labelled as PROPERTY-QUALIFIER, and two is labelled as a BEDROOM-QUANTIFIER. Structured aspects are obtained by matching *structured aspect patterns* (hence SAP) against free text. An SAP mirrors the structure of a structured aspect and consists of a tuple $\langle \mathsf{P}, \mathsf{T} \rangle$, where T is either the surface form of an aspect term or a POS tag, and $\mathsf{P}=\{\mathsf{p}_1, \ldots, \mathsf{p}_m\}$ is a sequence of surface forms of a modifier (e.g., two, mid terrace), POS tags (e.g., CD, JJ), or (nested) SAPs. An SAP is matched against free text in the usual way. For symmetry w.r.t. structured aspects, T and P are called the *head* and *tail* of the SAP respectively. A possible SAP matching the structured aspect above is $\langle \{\text{Victorian}, \langle \{\text{CD}\}, \text{bedroom} \rangle, \text{JJ terrace}\}, \text{property} \rangle$.

Overview and architecture SAPs are *induced* from a homogeneous corpus of texts, i.e., with texts coming from the same domain, e.g., restaurants, products. The induction operates in four phases: (1) normalization, (2) clustering, (3) segmentation and typing, and (4) generalization. Texts are processed to obtain a corpus of orthographically normalized and POS tagged noun phrases (NP). The normalized NPs are clustered around their head nouns, being these likely candidates for aspect terms. A cPMI-based (Damani and Ghonge, 2013) segmentation of the modifiers of the NPs locates *(i)* multi-word expressions and *(ii)* nested aspects. The segmented NPs are then taken as a first approximation of an SAP (ground), where the head noun becomes the head of the SAP and the (segmented) modifier of the NP becomes its tail. Modifiers are then typed as quantifiers or qualifiers. Ground SAPs are then generalized by replacing the surface forms of the modifiers by their POS tags, thus obtaining the final SAPs.

Normalization and POS tagging The normalization tackles issues that affect subsequent phases of the induction process. In particular, it improves the accuracy of noun chunking on NUT, avoiding an expensive (and possibly pointless) NUT-specific training. The first problem is the high number of tokenization errors, leading to incorrect sentence splits. To address this, we have built a NUT-specific tokenizer, producing correct tokens in the presence of, e.g., abbreviations, units of measures, and incorrect sentence boundaries. These customizations are domain independent. Incorrect orthography is another major problem in NUT. It directly affects the performance of POS taggers and, in turn, of noun chunking. To address this, we normalize the orthography of each token to its most common orthography in the whole corpus. We consider five orthographic classes: uppercase, lowercase, upper initial, lower initial, and alphanumeric. An occurrence of a token is normalized if more than a certain percentage (experimentally set at 90% across all domains) of its other occurrences in the corpus have a different orthography.

In the first sentence of Example 2, orthographic errors lead to wrong tags for, e.g., Fantastic (CD), Beds (NNPS), and OXFORD (VBD), while the missing space after the first full stop leads to an incorrect sentence boundary. The POS tagger cannot therefore produce the correct tags for Don't. The second sentence

Example 2: Normalization

Fantastic 2 Beds OXFORD property.Don ' t miss it !
CD CD NNPS VBD NN POS NN VBP PRP PUNC

↓

{fantastic 2 beds Oxford property . } {Do n't miss it ! }
JJ CD NN NNP NN PUNC VB RB VB PRP PUNC

shows how POS tagging improves after our improved tokenization and orthographic normalization. We do not only produce more sensible POS tags, but we also recover the correct sentence boundaries.

Normalized texts are then split into sentences and POS tagged. We use a state-of-the-art rule-based sentence splitter and Hepple's POS tagger tagger (Hepple, 2000) trained on the Penn TreeBank Corpus. The tagged sentences are handed over to a rule-based NP chunker (Ramshaw and Mitchell, 1999) to produce a corpus of noun phrases.

Noun-phrase clustering NPs are clustered around their head nouns. Head nouns are stemmed and lemmatized, e.g., by normalizing plurals, to avoid over-segmentation of the clusters due to different surface forms of equivalent head nouns. The modifiers of the NPs are also normalized to prevent non-content prefixes and numerical expressions from fragmenting the clusters.

NPs are generalized to abstract forms where non-content words and numerical expressions are replaced by POS tags. A non-content word is a token with a POS tag in {CC, DT, EX, IN, PRP, PUNC}, while a numerical expression has POS tag CD. Prefixes consisting only of non-content words are then removed from the NPs. The process is illustrated in Example 3.

Example 3: Clustering

Two further double bedrooms	CD further double bedrooms	{CD further double bedrooms, [BEDROOM]
Three further double bedrooms	CD further double bedrooms	further double bedrooms,
A further double bedroom	DT further double bedroom	CD first floor bedrooms}
Two first floor bedrooms	CD first floor bedrooms	

We start with four NPs with 2 different surface forms for the head noun (i.e., bedroom and bedrooms). For three of the NPs, the modifier starts with a numerical expression. This remains part of the NP but as a POS tag (CD). The non-content word prefix A (POS tag DT), is removed from the NP. The result of this process is a single cluster headed by BEDROOM and consisting of three partially-generalized NPs. The clusters are filtered based on their cardinalities, i.e., the number of (possibly duplicated) NPs belonging to the cluster. The top-k clusters whose elements cover at least a certain percentage (experimentally set at 70% across all domains) of all the NPs are retained. Clusters can also be fragmented by spelling errors. The head nouns of discarded clusters are therefore checked for similarity against the head nouns of the retained ones. Two clusters are merged if the Dameraau-Levenshtein string edit distance is less than an experimentally set threshold (20% across all domains). If multiple merging options are possible, the one with highest similarity is chosen. If only equivalent options are available, we merge in all possible ways. Clearly, the normalization of the noun-phrase modifiers described above can affect the ranking of the noun-phrases, since clusters can be merged thus increasing their ranking.

Segmentation and typing The segmentation phase identifies multi-word expressions and hierarchical structures in NP modifiers, thus producing a first approximation of an SAP. The key tool used in the segmentation is corpus-level significant point-wise mutual information (cPMI) (Damani and Ghonge, 2013). Our definition of cPMI uses the corpus of NPs instead of arbitrary descriptions. Let $\mathcal{C}$ be the set of all clusters produced as described above. We denote by $f_{\mathcal{C}}(\mathrm{t})$ the frequency of the string t in all clusters of $\mathcal{C}$, i.e., obtained by summing up all of the occurrences of t in all clusters. Let $0 < \delta < 1$ be the normalization factor defined as in (Damani and Ghonge, 2013), and t‖w, the concatenation of two strings

t and w. We then define $\text{cPMI}_{\mathcal{C}}(\mathsf{t}, \mathsf{w})$ as follows:

$$\text{cPMI}_{\mathcal{C}}(\mathsf{t}, \mathsf{w}) = \log \frac{f_{\mathcal{C}}(\mathsf{t}\|\mathsf{w})}{f_{\mathcal{C}}(\mathsf{t}) \cdot \frac{f_{\mathcal{C}}(\mathsf{w})}{|\mathcal{C}|} + \sqrt{f_{\mathcal{C}}(\mathsf{t})} \cdot \sqrt{\frac{ln(\delta)}{(-2)}}}$$

The cPMI value is used to determine whether a token should be associated with *(i)* the head noun, *(ii)* a nested token representing the head of a different cluster, thus possibly inducing a nested structure, or *(iii)* an adjacent token, thus forming a multi-word expression.

We model the segmentation as the problem of finding a cPMI-optimal *parenthesization* (or, equivalently, a parse tree) of the NP. In order to achieve this, we first have to define the notions of *(i)* valid parenthesization and *(ii)* cPMI of a parenthesization. A valid parenthesization is a balanced parenthesization such that, each k-th level of the parenthesization: **(1)** consists of at least two elements (i.e., tokens or subpatterns), and **(2)** it either terminates with a head of cluster or it contains no heads of cluster. The cPMI of a (valid) parenthesization is the sum of the cPMI values computed between each element (token or parenthesized sub-expression) in the parenthesization and the first head of cluster following the element at the same level of nesting. If no heads of cluster are present at the same level of nesting, then the sum of the cPMI values between subsequent tokens is taken. The cPMI of the parenthesized sub-expressions is then recursively added to this value.

Example 4: Segmentation

p	:	Victorian two bedroom mid terrace property
p_i	:	(Victorian (two bedroom mid) (terrace) property)
p_v	:	(Victorian (two bedroom) (mid terrace) property)
SAP	:	$\langle\{$Victorian, $\langle\{$two$\}$, bedroom$\rangle$, mid terrace$\}$, property$\rangle$

Consider the NP p of Example 4. A balanced but invalid parenthesization is shown as p_i. The parenthesization violates both conditions given above, i.e., the level-2 parenthesization (two bedroom mid) contains a head of cluster (i.e., bedroom) but terminates with the token mid, and the level-2 parenthesization (terrace) contains a single token. A valid parenthesization is p_v and its cPMI is computed as:

$$\begin{aligned}\text{cPMI}_{\mathsf{np}} \;=\; & \text{cPMI}_{\mathcal{C}}(\text{Victorian, property}) \;+\; \text{cPMI}_{\mathcal{C}}(\text{two bedroom, property}) \;+ \\ & \text{cPMI}_{\mathcal{C}}(\text{mid terrace, property}) \;+\; \text{cPMI}_{\mathcal{C}}(\text{two, bedroom}) \;+\; \text{cPMI}_{\mathcal{C}}(\text{mid, terrace})\end{aligned}$$

Notice that $\text{cPMI}_{\mathsf{np}}$ now includes the cPMI value between mid and terrace (i.e., a multi-word expression).

Algorithm 1 formalizes the process described above. The pseudocode focuses on the most technical aspect of the procedure, i.e., the generation of the different parenthesizations. We also omit minor technicalities related to, e.g., handling of punctuation and non-content words. The algorithm receives as an input the clustered noun phrases, here represented as a map $\mathcal{C}$ having as keys the heads of clusters and as values (denoted as `val`$(\mathcal{C})$) sets of noun phrases. The output of the algorithm is again a map P, having as keys the heads of the patterns, and sets of SAPs as values. The algorithm iterates over all noun phrases np in all clusters and converts them into a sequence of tokens P (Line 4). The function `optiPar` then produces a cPMI optimal parenthesization of P (Line 5) that is then transformed into an SAP via the `par2SAP` function (Line 6). The SAP is stored in P (Line 7). The notation P`(`k`)` represents the cluster of SAPs headed by k, while `head(`P`)` returns the head of the pattern P. The function `optiPar` is an adaptation of standard combinatorial parenthesization algorithms, where we keep track of the parenthesization producing the highest noun-phrase-wide cPMI. The function takes a sequence of tokens P and recursively parenthesizes it (Lines 4-9). The $\text{cPMI}_{\mathsf{np}}$ of the parenthesization is then computed by the function `npcPMI` and tested against the current maximum (Lines 11-13). The function `isValid` in Line 10 checks that only valid parenthesizations are considered. For the sake of readability, Algorithm 1 gives a straightforward recursive implementation of the procedure but more efficient dynamic programming implementations can be provided.

In SAPs, nested patterns, e.g., $\langle\{$two$\}$, bedroom$\rangle$ are replaced with a reference to the cluster of SAPs headed by the head of the nested pattern, e.g., BEDROOM. The reason for this is to enable parallel matching

Algorithm 1: Segmentation

```
Function segmentation(C)
    input: C: the clusters
    P ← ∅;
    for C ∈ val(C)
        for distinct np ∈C
            P ← split(np);
            P_max ← optiPar(P,C);
            P_sap ← par2SAP(P_max);
        P(head(P_sap)) ← P(head(P_sap)) ∪ {P_sap};
    return P;
Function optiPar(P,C)
    input: C: the clusters
    input: P: the sequence of tokens from the noun phrase
    if |P| ≤ 2 then
        return P;
    ⟨max_pmi, P_max⟩ ← ⟨0, ∅⟩;
    for i ∈ [0, |P|-1]
        for j ∈ [|P|-1, i+1]
            if i > 0 ∨ j < |P|-1 then
                P′ ← optiPar(sub(P,0,i)) || '(' || optiPar(sub(P,i,j)) || ')' || optiPar(sub(P,j,|P|-1));
            else
                P′ ← '(' || P || ')';
            if isValid(P′) then
                cur_pmi ← npcPMI(P′,C);
                if cur_pmi > max_pmi then
                    ⟨max_pmi, P_max⟩ ← ⟨cur_pmi, P′⟩;
    return P_max;
```

of the SAP at runtime, thus avoiding matching multiple times the same fragment of text. The elements of a ground SAP are then *typed* according to their role in the pattern (Example 5). The head of the pattern is labelled as the head of the SAP cluster it belongs to and is used to match the aspect term. Modifiers with a POS tag CD and linked to an aspect term t are labelled as λ(t)-QUANTIFIER. All other modifiers linked to t, including nested SAPs, are labelled as λ(t)-QUALIFIER. Extractions produced by matching SAP against free text are labelled according to the typing of the SAP.

Example 5: Typing

⟨{Victorian, ⟨{two}, bedroom⟩, mid terrace}, property⟩

property	→	PROPERTY	bedroom	→	BEDROOM
Victorian	→	PROPERTY-QUALIFIER	two	→	BEDROOM-QUANTIFIER
mid terrace	→	PROPERTY-QUALIFIER			
two bedroom	→	PROPERTY-QUALIFIER			

Generalization Ground SAPs are limited to what has been directly observed in the corpus. As a consequence, the elements of an SAP are generalized to their POS tags to increase recall. The process must be controlled because excessive generalization may also lead to inaccurate extractions. The generalization rules are illustrated in Example 6 and operate as follows: (1) Numerical expressions, non-content words, and punctuation are always generalized to their POS tags. (2) Multi-word expressions are generalized as both n-grams and unigrams since they "behave" like a single token. In the latter case, the POS tag of the last token is used. (3) Aspect terms are not generalized unless the tail of the pattern contains a nested SAP with a ground head. The nature of the modifiers is often strictly related to the aspect term they refer to. This is obvious for, e.g., the modifiers of bedroom and property in Example 6. We therefore have to prevent undesired associations between aspect terms and incompatible modifiers. (4) Qualifiers are also semantically related to other qualifiers at the same level of nesting. As a consequence, we generalize at most one qualifier per level of nesting to maintain this connection. Clearly, this process may give rise to more than one generalization for the same SAP. The ground pattern is also preserved.

Example 6: Generalization

⟨{Victorian, ⟨{two}, bedroom⟩, mid terrace}, property⟩

↓

⟨{JJ, ⟨{CD}, bedroom⟩, mid terrace}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, JJ}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, JJ terrace}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, mid JJ}, property⟩
⟨{Victorian, ⟨{CD}, bedroom⟩, mid terrace}, NN⟩

Patterns are scored based on their ability to discriminate between correct and incorrect extractions. We adapt the scoring mechanism of (Gupta and Manning, 2014), where the quality of a pattern is estimated by matching the extracted patterns against a manually labelled validation set. In our unsupervised setting, no labelled dataset is available. We take the heads of the noun-phrase clusters as a surrogate of the set of valid aspects. The analysis is limited to aspect terms. Let T be the set of valid aspect terms as defined above, and E be the set of aspect terms produced by an SAP P. The score of P is computed as:

$$\nu(\mathsf{P}) = \frac{|T|}{\sum_{e \in E}(1 - \max_{t \in T} \mathbb{1}(\frac{dist(t,e)}{len(t)} < 0.2))} \cdot \log |T| \qquad \nu(\mathsf{P}) \in [0, \infty]$$

where $dist(t, e)$ denotes the Dameraau-Levenshtein edit distance between two strings t and e and $len(\cdot)$ denotes the length of the string. Patterns scoring less than an experimentally set threshold (50% across all domains) are eliminated.

3 Evaluation

Our method (Oxtractor) is implemented in Java. All experiments are run on a quad-core desktop machine at 3.40GHz and 32GB RAM, running Linux. All resources used in the evaluation are made available for replicability at http://bit.ly/2dewzdd and include: the SAED dataset and GS, our reimplementations of IIITH and ATL, a compiled version of Oxtractor, and all output files generated by all systems.

Datasets and metrics We use three groups of datasets in our evaluation (Table 1): The first two consist of the SemEval14 and SemEval15[2] datasets used for the aspect term extraction (ATE) and opinion target expression (OTE) subtasks of the aspect-based sentiment analysis (ABSA) task. The datasets provide laptops, restaurants and hotel reviews with associated gold standard (GS) annotations. The hotel domain is meant to be used in a completely unsupervised setting and therefore no training data is available. The size of SemEval14 in Table 1 is expressed in number of sentences instead of number of texts since this information is unavailable. We complement the SemEval15 datasets with some specifically designed for SAE (SAED). We provide texts from six domains (50% for testing, 50% for validation). Four of them, i.e., chairs, real estate, shoes, and watches, describe products. The Amazon texts come from the Stanford's Snap Lab's web data corpus (McAuley and Leskovec, 2013). The two remaining domains, i.e., hotels and restaurants, can be classified as services. These descriptions are still feature intensive but, differently from the products, the features are loosely connected to the main entity, i.e., locations, services/facilities offered. The dataset consists of both NUT and (semi-) formal English texts. We provide GS annotations for 150 texts equally distributed across the six domains. The GS provides and average of 355 aspect terms, 30 quantifiers, 430 qualifiers, and 45 nested aspects per domain. Annotations were produced by 6 independent annotators (λ=87%). We use standard recall, precision, and F_1 score metrics. However, due to the different granularity of the output produced by the systems and of the GS annotations, the definition of a correct extraction varies slightly with each evaluation task.

Comparative evaluation – Simplified SAE The method by (Kim et al., 2012), hence ATL, is currently the closest to SAE we are aware of. We have obtained from the authors the dataset used in their evaluation but not an implementation of the system. We have reimplemented the method and successfully reproduced the experimental results described in the original paper. Figure 1 shows a comparison between ATL and Oxtractor on the SAED dataset. An extraction is correct if modifiers and aspect terms match exactly the GS annotations, and if modifiers are correctly typed as qualifiers or quantifiers. This is a simplified

Table 1: Datasets

DATASET	DOMAIN	SIZE (#texts)	SOURCES	CATEGORY	FORMALITY	TYPE
SemEval14	restaurants	3k + 800 GS (*)	Citysearch	service	NUT	evaluative
	laptops	3k + 800 GS (*)	N/A	product	NUT	evaluative
SemEval15	restaurants	254 + 96 GS	Citysearch	service	NUT	evaluative
	hotels	N/A + 30 GS	Citysearch	service	NUT	evaluative
SAED	chairs	94k + 25 GS	Amazon, GumTree	product	NUT	descriptive
	hotels	20k + 25 GS	TripAdvisor	service	formal	descriptive
	real estate	87k + 25 GS	RightMove	product	semi-formal	descriptive
	restaurants	115k + 25 GS	TripAdvisor	service	formal	descriptive
	shoes	46k + 25 GS	Amazon, GumTree	product	NUT	descriptive
	watches	10k + 25 GS	Amazon, GumTree	product	NUT	descriptive

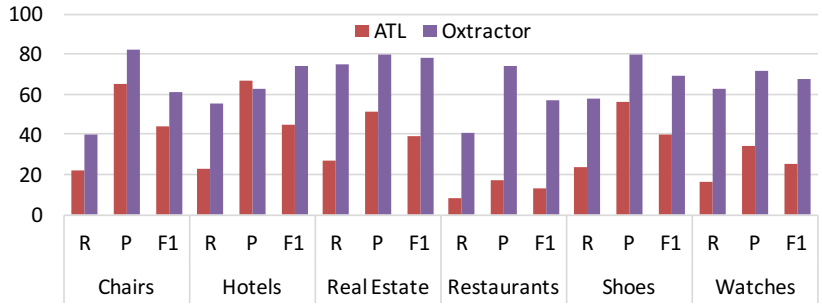
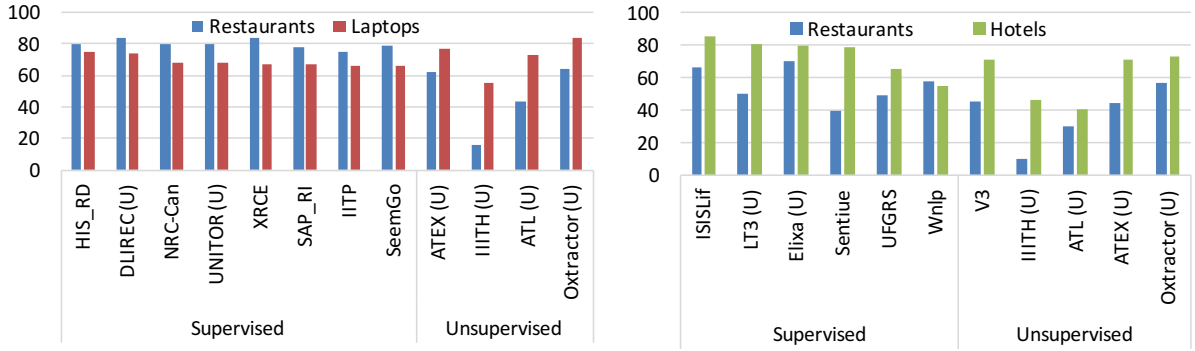


Figure 1: Oxtractor vs. ATL on simplified SAE (SAED dataset)

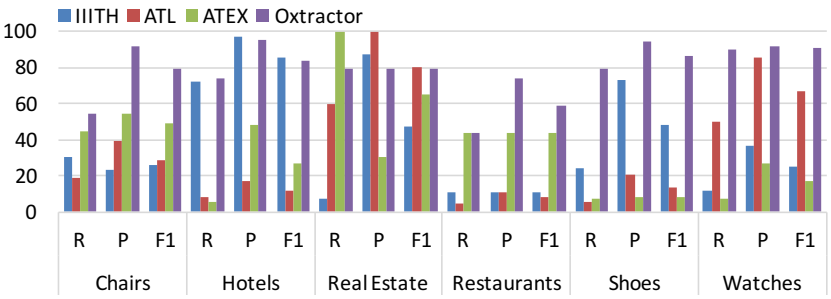
SAE setting where we do not require correct linking of modifiers to aspect terms. Oxtractor performs 33% better than ATL in average, outperforming it in all domains. Besides being unable to extract hierarchical structures, a visible issue in ATL is the inability to recognize semantic connections between modifiers and aspect terms. This leads to a number of incorrect extractions for both aspect terms and modifiers that could be avoided by leveraging, e.g., statistical co-occurrence or cPMI.

Comparative evaluation – ATE Restricting the evaluation to aspect terms makes it possible to compare Oxtractor against other ATE systems. We denote by IIITH and ATEX the methods proposed by (Raju et al., 2009) and (Zhang and Liu, 2014) respectively. IIITH uses unsupervised clustering of noun-phrases to derive aspect terms and is therefore similar to Oxtractor. We could not obtain the original IIITH system from the authors so the evaluation relies on our own implementation. ATEX, on the other end, is chosen because of its ATE method based on topic modeling. Moreover, it is freely available for testing. An aspect term is correctly extracted if it matches exactly a GS annotation. For Oxtractor, IIITH, and ATL we used the SAED corpora for training. Figures 2a and 2b show the results for the SemEval14 and SemEval15 datasets respectively. For all systems, except Oxtractor, IIITH, ATEX, and ATL, we report the scores provided in the corresponding SemEval papers. The symbol (U) denotes systems that have used additional data besides the training set provided by SemEval (i.e., *unconstrained* in SemEval terminology). Oxtractor outperforms all unsupervised systems and some of the supervised ones. Moreover, this is a lower bound for Oxtractor due to a difference between the granularity of the SemEval GS and the output produced by Oxtractor. E.g., Egyptian restaurant is considered a correct aspect term by SemEval but Oxtractor would only produce restaurant as the aspect term and Egyptian as its modifier (and thus a miss for SemEval). A striking result is the performance achieved by Oxtractor on the laptops domain in SemEval14, where also all supervised systems are outperformed. As for many other product-like domains, aspect terms in the laptops domain frequently fall within the scope of noun-phrases that are easily processed by our method. This is much less true for other service-like domains such as, e.g., restaurants and hotels. Figure 2c shows the performance of Oxtractor, IIITH, ATEX, and ATL on the SAED dataset. In this case, the GS differentiates between aspect terms and modifiers, thus explaining the lower performance of traditional ATE systems, e.g., IIITH and ATEX, and the higher accuracy of, e.g., ATL that is able to appreciate this difference. ATEX

[2]http://alt.qcri.org/semeval2014/task4/ and http://alt.qcri.org/semeval2015/task12/



(a) SemEval14 Dataset (b) SemEval15 Dataset



(c) SAED Dataset

Figure 2: Oxtractor vs. others in ATE

also struggles on restaurants due to long sentences.

Full SAE We evaluate the performance of Oxtractor in the full SAE setting using the SAED dataset. An extraction is correct if aspect terms and modifiers match the GS annotations, including the correct (and possibly hierarchical) associations between modifiers and entities. In average (Figure 3), Oxtractor achieves an F_1 of 58.6% in the full SAE setting, sensibly below the one obtained in the ATE (i.e., 79.6%) and simplified SAE (i.e., 67.8%) settings. Linking modifiers to aspect terms in the presence of hierarchical structures is indeed a much more challenging task than simply identifying them. Another interesting aspect is the impact of the generalization on the performance. Generalized SAPs produce 444 correct extractions against the 386 of the ground ones (+15%).

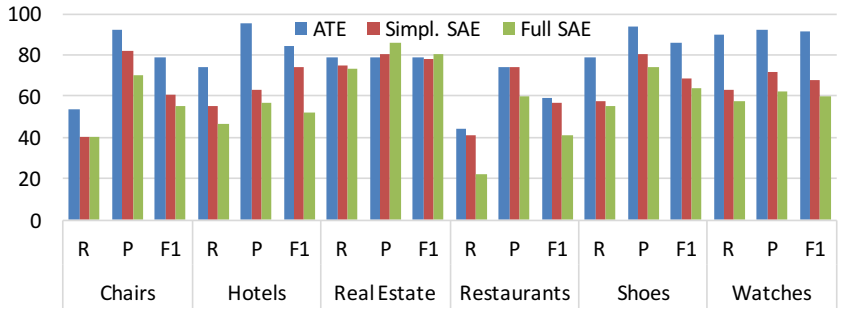
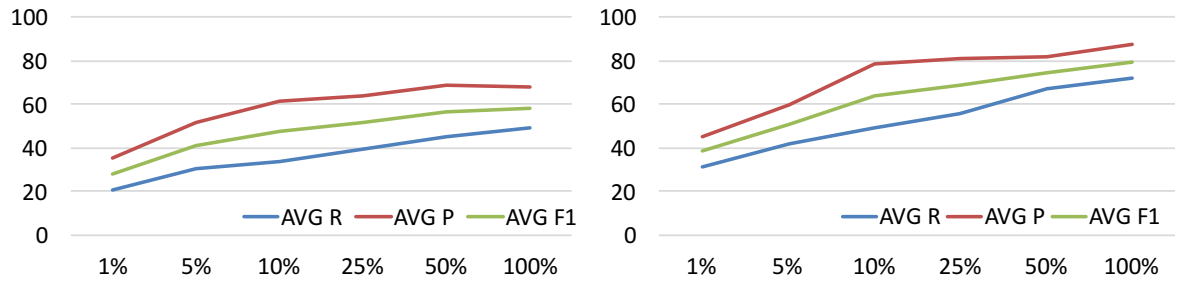


Figure 3: Oxtractor on full SAE

Corpus size One obvious question is how dependent our method is on the size of the training corpus. To measure this, we evaluated Oxtractor by inducing SAPs from increasing subsets of the original corpora corresponding to fractions of 1%, 5%, 10%, 25%, and 50% of their original size. Figure 4 shows the effect of the corpus size on the performance of Oxtractor in both the SAE and ATE settings. Clearly, larger corpora lead to better results in both settings. However, two interesting facts have been observed. Long-

tail aspects are only induced from sufficiently large corpora (thus the behavior between 10% and 50%). Larger corpora also have the disadvantages that sufficiently frequent but incorrect tokens can end up being extracted as modifiers. In other words, the increase in recall is not matched by a comparable increase of precision (Figure 4b). In the case of SAE we even observe a slight drop in precision (Figure 4a).



(a) SAE Task (b) ATE Task

Figure 4: Performance vs. corpus size (average – SAED dataset)

A breakdown of the data per domain allows us to draw further conclusions on the relationship between the size of the corpus and the performance of the SAPs. There is a relationship between the variety of features and the amount of data that is necessary to induce good quality SAPs. For domains such as, e.g., chairs, realestate, shoes, and watches, starting from 25% of the size of the corpus we do not notice substantial improvements in performance. This can be explained by the nature of the features in these domains that are intrinsically limited, e.g., make and models of the products, types of real estate properties, etc. In the restaurants and hotel domains the texts are much more variegated in features, e.g., restaurant and hotel names, dishes, locations, etc. Despite the large amount of texts available, it seems that our method would require even larger corpora before being able to converge to a stable set of aspects.

Efficiency Finally, we evaluate the efficiency of the SAP induction and matching phases. Oxtractor's efficiency mostly depends on the length of the sentences, due, e.g., to the morphological analysis, our cPMI-based segmentation, and pattern matching. Oxtractor induces SAPs at a rate of 14 $^{ms}/_{sent}$ and 6 $^{ms}/_{sent}$ for *long* (i.e., $\geq$10 tokens) and *short* (i.e., $<$10 tokens) sentences respectively. The matching time per sentence is almost negligible and ranges between $2ms$ and $3ms$ per text. We also notice a linear correlation between the size of the SAP and its matching time. This is achieved, despite the presence of hierarchical structures, by replacing nested patterns with references to the corresponding SAP clusters, enabling parallel matching of the nested SAPs. In terms of training time, Oxtractor induces patterns from $20k$ texts within $1hr$ on average. IIITH and ATL require more than $15hrs$ on the same dataset.

Discussion SAE is still in its infancy. A number of interesting problems can be studied in this area such as, e.g., semantic categorization of modifiers and aspect terms. Although the majority of structured aspects appear in noun phrases, a fair amount also appears in more complex syntactic structures. We extended our normalization to rewrite these structures into traditional noun-phrases with good results. Another issue is redundant SAPs, caused by aspect terms having the same "tail" and semantically belonging to a taxonomic hierarchy of concepts, e.g., two bedroom apartment, two bedroom house. We are currently investigating the use of knowledge bases such as, e.g., BabelNet (Navigli and Ponzetto, 2012) to reduce this redundancy. Finally, our evaluation shows that the gap between Oxtractor and supervised systems is still considerable. Oxtractor can be adapted to use supervision at different stages of the induction process, in particular during clustering and pattern scoring.

Acknowledgements

The research leading to these results has received funding from the EPSRC Programme Grant VADA, no. EP/M025268/1, and the ERC Grant ExtraLytics, no. ERC-2014-PoC. Oxtractor is not related to any of the commercial products currently offered by Meltwater.

References

Justine Cassell. 2000. Embodied conversational interface agents. *Commun. ACM*, 43(4):70–78.

Zhiyuan Chen, Arjun Mukherjee, and Bing Liu. 2014. Aspect extraction with automated prior knowledge learning. In *Proc. of ACL*, pages 347–358.

Yejin Choi and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proc. of ACL*, pages 269–274.

Om P Damani and Shweta Ghonge. 2013. Appropriately incorporating statistical significance in pmi. In *Proc. of EMNLP*, pages 163–169.

R. Ghani, K. Probst, Y. Liu, M. Krema, and A. Fano. 2006. Text mining for product attribute extraction. *SIGKDD Explorations*, 8(1):41–48.

Sonal Gupta and Christopher D Manning. 2014. Improved pattern learning for bootstrapped entity extraction. In *CoNLL*, pages 98–108.

M. Hepple. 2000. Independence and commitment: Assumptions for rapid training and execution of rule-based pos taggers. In *Proc. of ACL*, pages 278–277.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proc. of SIGKDD*, pages 168–177.

Niklas Jakob and Iryna Gurevych. 2010. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proc. of EMNLP*, pages 1035–1045.

Anitha Kannan, Inmar E Givoni, Rakesh Agrawal, and Ariel Fuxman. 2011. Matching unstructured product offers to structured product specifications. In *Proc. of SIGKDD*, pages 404–412.

C. Kelly, B. Devereux, and A. Korhonen. 2012. Semi-supervised learning for automatic conceptual property extraction. In *Proc. of CMCL*, pages 11–20.

D. S. Kim, K. Verma, and P. Z. Yeh. 2012. Building a lightweight semantic model for unsupervised information extraction on short listings. In *Proc. of EMLNP*, pages 1081–1092.

Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proc. of ACL*, pages 653–661.

Kang Liu, Liheng Xu, and Jun Zhao. 2013. Syntactic patterns versus word alignment: Extracting opinion targets from online reviews. In *Proc. of ACL*, pages 1754–1763.

J. McAuley and J. Leskovec. 2013. Hidden factors and hidden topics: Understanding rating dimensions with review text. In *Proc. of RecSys*, pages 165–172.

R. Navigli and S. P. Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artif. Intell.*, 193:217–250.

Bo Pang and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135.

Ana-Maria Popescu and Oren Etzioni. 2005. Extracting product features and opinions from reviews. In *Proc. of HLT-EMNLP*, pages 339–346.

Soujanya Poria, Erik Cambria, Lun-Wei Ku, Chen Gui, and Alexander Gelbukh. 2014. A rule-based approach to aspect extraction from product reviews. In *Proc. of COLING*, pages 28–37.

K. Probst, R. Ghani, M. Krema, A. E. Fano, and Y. Liu. 2007. Semi-supervised learning of attribute-value pairs from product descriptions. In *Proc. of IJCAI*, pages 2838–2843.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

S. Raju, P. Pingali, and V. Varma. 2009. An unsupervised approach to product attribute extraction. In *Proc. of ECIR*, pages 796–800.

L. A. Ramshaw and M. P. Mitchell. 1999. Text chunking using transformation-based learning. In Armstrong S. et Al, editor, *Natural Language Processing Using Very Large Corpora*, volume 11 of *Text, Speech and Language Technology*, pages 157–176. Springer.

Christina Sauper and Regina Barzilay. 2013. Automatic aggregation by joint modeling of aspects and values. *JAIR*, 46(1):89–127.

Jaeho Shin, Sen Wu, Feiran Wang, Christopher De Sa, Ce Zhang, and Christopher Ré. 2015. Incremental knowledge base construction using DeepDive. *PVLDB*, 8(11):1310–1321.

Ivan Titov and Ryan T McDonald. 2008. A joint model of text and aspect ratings for sentiment summarization. In *Proc. of ACL*, volume 8, pages 308–316.

Yuanbin Wu, Qi Zhang, Xuanjing Huang, and Lide Wu. 2009. Phrase dependency parsing for opinion mining. In *Proc. of EMNLP*, pages 1533–1541.

Mohamed Yahya, Steven Whang, Rahul Gupta, and Alon Y Halevy. 2014. Renoun: Fact extraction for nominal attributes. In *Proc. of EMNLP*, pages 325–335.

Bishan Yang and Claire Cardie. 2014. Joint modeling of opinion expression extraction and attribute classification. *TACL*, 2:505–516.

Jianxing Yu, Zheng-Jun Zha, Meng Wang, and Tat-Seng Chua. 2011. Aspect ranking: identifying important product aspects from online consumer reviews. In *Proc. of HLT*, pages 1496–1505.

Daniel Zeng, Hsinchun Chen, Robert Lusch, and Shu-Hsing Li. 2010. Social media analytics and intelligence. *IEEE Intell. Syst.*, 25(6):13–16.

Lei Zhang and Bing Liu, 2014. *Aspect and Entity Extraction for Opinion Mining*, pages 1–40. Springer Berlin Heidelberg.

Xinjie Zhou, Xiaojun Wan, and Jianguo Xiao. 2013. Collective opinion target extraction in chinese microblogs. In *Proc. of EMNLP*, pages 1840–1850.

Li Zhuang, Feng Jing, and Xiao-Yan Zhu. 2006. Movie review mining and summarization. In *Proc. of CIKM*, pages 43–50.