A method for automatic extraction of multiword units representing business aspects from user reviews[*]

Olga Vechtomova

Department of Management Sciences, University of Waterloo
200 University Avenue West, Waterloo, Ontario, N2L 3GE, Canada
Tel: +1 519 888 4567 ext. 32675; Fax: +1 519 746 7252
Email: ovechtom@uwaterloo.ca

Abstract

The paper describes a semi-supervised approach to extracting multiword aspects of user-written reviews that belong to a given category. The method starts with a small set of seed words representing the target category, and calculates distributional similarity between the candidate and seed words. We compare three distributional similarity measures (Lin's, Weeds' and balAPinc), and a document retrieval function, BM25, adapted as a word similarity measure. We then introduce a method for identifying multiword aspects by using a combination of syntactic rules and a co-occurrence association measure. Finally, a method for ranking multiword aspects by the likelihood of belonging to the target aspect category is described. The task used for evaluation is extraction of restaurant dish names from a corpus of restaurant reviews.

1. Introduction

There is a rapidly growing trend of end-users writing reviews of businesses, services, products and events on the Web. Numerous websites, for example CityGrid and TripAdvisor, allow people to write reviews of businesses, such as restaurants and hotels. While many of them offer different search features, such as search by location or by the opinion score, there is a need for a more fine-grained search by different aspects of businesses that are discussed in user reviews. For instance, a user may be interested to know which restaurants serve the best seafood dishes, or the user may like to see the list of dish names that people have commented on positively in their reviews of a particular restaurant. As the number of reviews for businesses and products is constantly growing, it is becoming increasingly time-consuming for a user to read through all of them. It would be, therefore, useful to automatically extract aspects of businesses or products that people comment about in their reviews. The initial motivation for the work presented in this paper stemmed from a real need of our industry partner in a recent research project, which was developing a system for recommending local activities, such as eating out, to users, and needed a method for automatically extracting aspects of specific categories from user-written reviews, so that they could be included in the recommendations of a specific business.

In this paper we present a complete system for extracting multiword units (MWUs) belonging to a given category of business or product aspects. The method is semi-supervised: given a small number of seed words representing a specific category of aspects, for example, dish names in restaurant reviews, we identify other dish names that people mentioned in their reviews. Identifying review aspects belonging to a specific category has different challenges compared to a general task of building lexicons of semantically similar words. Firstly, many of the aspects are

[*] Published in JASIST, 65(7), pp.1463–1477.

multiword units. Secondly, many of them have complex syntactic structure, including prepositions and conjunctions, for example: “roasted asparagus with pancetta, parmesan, olive oil and a fried egg on top”. Thirdly, many are rare or even unique, i.e. occur only once in the corpus of reviews or even the Web. Lastly, many aspects are pre-modified by subjective adjectives, which must be identified and separated. We propose a multi-staged approach towards identifying such units that uses a combination of statistical methods, such as distributional similarity measures and co-occurrence association measures, and Natural Language Processing (NLP) techniques such as syntactic dependency parsing. The presented system could be used in a variety of end-user applications, for example a recommender system that suggests a restaurant and shows to the user the list of dish names mentioned in other users’ reviews. Another example is an application that calculates a score for a restaurant based on the polarity of opinions expressed about different categories of review aspects, e.g. dish names, ambiance and staff.

The advantage of a semi-supervised approach compared to unsupervised methods, such as clustering, is that there is more control over what categories of entities are extracted for a particular application. For instance, in restaurant reviews, there may be a need to identify dish names, aspects related to ambiance and atmosphere, and aspects related to service and staff. Alternatively, there may be a need to identify more fine-grained categories of dish names, such as desserts or beverages. By providing a small set of seed words that belong to the desired category, the end-user or the application developer can control what aspects are extracted. Unsupervised methods, such as clustering, do not offer such control, and are more suited for applications where there is a need to discover the categories of aspects that exist in the corpus.

On the other hand, the advantage of semi-supervised approaches over supervised methods, is that the latter require a large set of labeled data, for instance, all dish names labeled in a large subset of a corpus. Manual labeling of a large dataset is a labour-intensive task. In contrast, semi-supervised methods only require a small set of example seed words, which can be easily provided by application developers or even end-users.

The presented system was evaluated on a corpus of restaurant reviews. The task used for evaluation is extraction of dish names. Dish names are a particularly challenging category of aspects. There are many long and syntactically complex MWUs among them. Also, many MWU dish names have low frequency, which makes it challenging to apply some statistical methods commonly used for calculating similarity between words. For instance, we cannot effectively apply distributional similarity methods on the whole MWUs with low frequencies, since there is not enough context to calculate similarity between them and seed words.

The major novel contribution of the work described in this paper compared to our two earlier papers (Vechtomova and Robertson, 2012; Vechtomova, 2012) is the method for identifying and weighting syntactically complex MWUs, which belong to a specific category (Section 2.3). In (Vechtomova and Robertson, 2012) the method for using BM25 as a term-term similarity measure was proposed, and evaluated on the Entity track and QA datasets of TREC. In (Vechtomova, 2012), this measure was applied to the task of dish name extraction, however, the method presented there used only simple noun phrases conforming to the “JJ(≥0) NN(≥1)” pattern, output by a Noun Phrase chunker. This paper presents new methods for identifying and weighting syntactically complex MWUs (Sections 2.3 and 3.2). The weighting of single words by distributional similarity to seeds (Section 2.2) is done using the method presented in (Vechtomova, 2012), therefore here we used the tuning parameters determined experimentally in that work.

The paper is organised as follows: Section 2 describes the proposed system; Section 3 presents evaluation; Section 4 gives an overview of related work, and Section 5 concludes the paper and outlines future research directions.

2. Methodology

2.1 Architecture of the system

The overall architecture of the system is presented in Figure 1. The goal of the system is to extract from each review all MWUs that belong to the target category of aspects. To illustrate the use of the system, we use restaurant dish names as an example of the target category throughout the paper.

The system takes as input the corpus of reviews and a small initial set of seed words. The "Compute similarity" module (box 1b in Fig. 1) calculates distributional similarity between the set of dish name seed words and single nouns. One of the extensions of the system ("NPMI-weighted method" described in Section 2.3.1.2) additionally uses a small set of seed subjective adjectives to find and remove subjective modifiers in MWUs (box 1a in Fig. 1). In both modules 1a and 1b, the similarity is calculated between the words' feature vectors, where the features of a word are grammatical dependency triples it co-occurs with. An example of a dependency triple is "eat VB:dobj:NN pizza", which represents a direct object grammatical relationship between the verb "eat" and the noun "pizza". We compare a number of distributional similarity measures, namely, BM25 (Robertson et al., 1995) adapted to the use as a term-term distributional similarity measure (Vechtomova and Robertson, 2012), Lin's measure (Lin, 1998), a measure by Weeds and Weir (2003) and a directional similarity measure balAPinc by Kotlerman et al. (2009). The process of computing similarity between single words and seeds using BM25 is described in Section 2.2. An overview of the other three distributional similarity methods is given in Section 4.1.

The module "Identify and weight MWU aspects" (Box 2 in Fig. 1) takes a corpus of reviews parsed by a dependency parser, and uses a combination of syntactic rules and a co-occurrence association measure to identify MWU aspects. It also calculates the score for each MWU of how likely it is to belong to the target aspect category. In calculating the score, we use similarity of its constituent single words to the seeds, computed by module 1b. Section 2.3 describes the process of identifying and weighting MWU aspects in detail.

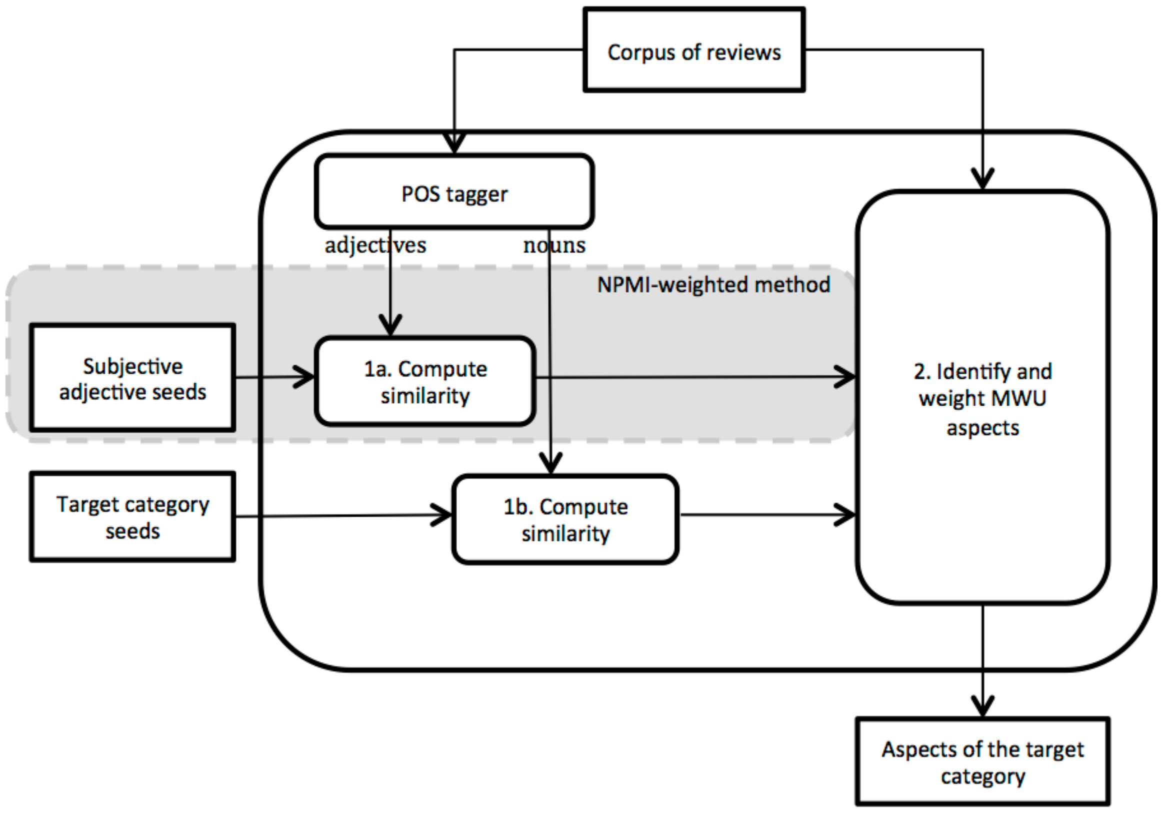

Figure 1. Architecture of the system

2.2 Ranking single words by distributional similarity to seeds

We start the process with a small set of seeds, which in our example task are single nouns denoting dish names, and a set of words we want to rank (candidate words), that comprise all single nouns in the corpus. In this section we describe the process of building feature vectors for both seeds and candidate words, and the process of ranking candidates with respect to seeds by using the BM25 function.

2.2.1 Building feature vectors

The context of each seed and candidate word is represented as a vector of context features. As a context feature we use grammatical dependency triples, in a similar way as was done by Lin (1998). In detail, our method consists of the following steps:

1. Perform dependency parsing of the corpus using Stanford dependency parser (de Marnefe et al., 2006). Each dependency triple consists of two words, their POS tag and a dependency relationship that connects them, for example: "eat VB:dobj:NN pasta".

2. For each seed word *s*, extract all dependency triples that contain it, and build a vector feature from each triple, by substituting the word *s* with "X". For instance, if we build a vector for the word "pasta", the dependency triple "eat VB:dobj:NN pasta" is transformed into: "eat VB:dobj:NN X". The method has a tuning constant *t*, which represents the number of seed words that a feature has to co-occur with in order to be included in the vector. The set {F} contains all features that have the seed co-occurrence frequency greater than *t*. Only these features are included in the vectors of seed words. The optimal values for *t* for each of the four measures

(BM25, Lin's method, Weeds' method and balAPinc) were experimentally determined in the previous work (Vechtomova, 2012), and are used in the experiments presented in this work.

3. For each candidate word *c* extract all dependency triples with which it co-occurs, transform them into features in the same way as in Step 2. Include in the vector only the features that belong to set {F}.

4. For each feature in the vector of a seed or candidate word, record TF, which is the frequency of co-occurrence of the feature with this word in the corpus. In the example above, TF of the feature "eat VB:dobj:NN X" in the vector of the word "pasta" is the frequency of occurrence of "eat VB:dobj:NN pasta" dependency triple in the corpus. We used the parsed corpus of 157,865 restaurant reviews to get frequencies of triples.

2.2.2 Computing similarity between vectors

The objective of this step is to rank all candidate words in the corpus by similarity to all seed words. The first step is to calculate similarity in a pairwise manner between each seed and each candidate word. The result of this step is a ranked list of candidates for each seed. The second step is combining these lists into one ranked list.

In this section we describe a method using BM25 as a term-term similarity function, presented in (Vechtomova and Robertson, 2012). In this method, similarity between vectors of a seed and a candidate is computed by using the BM25 document ranking model with query weights, called Query Adjusted Combined Weight (QACW) (Spärck Jones et al., 2000). In the QACW formula, the vector of the seed word *s* is treated as the query and the vector of the candidate word *c* as the document:

$$QACW_{c,s} = \sum_{f=1}^{F} \frac{TF(k_1 + 1)}{K + TF} \times QTF \times IDF_f \qquad (1)$$

Where: F – the number of features that a candidate word c and a seed word s have in common; TF – frequency of feature f in the vector of candidate word; QTF – frequency of feature f in the vector of the seed entity (computed in the same way as TF); $K = k_1 \times ((1-b)+b \times DL/AVDL)$; k_1 – feature frequency normalisation factor; b – document length normalisation factor; DL – number of features in the vector of the candidate entity; $AVDL$ – average number of features in all candidate entities.

The optimal values for b and k_1 for the task of ranking single dish names were determined experimentally in (Vechtomova, 2012) to be respectively 0.9 and 1.6. The same values are used in this work.

The *IDF* (Inverse Document Frequency) of the feature is calculated as follows:

$$IDF_f = log\frac{N}{n_f} \qquad (2)$$

Where, n_f – number of candidate word vectors the feature f occurs in, N – number of candidate word vectors.

After all candidate words are ranked by similarity to each seed, their scores in each ranked list are normalised so that they are in the range between zero and one. The normalised scores of the candidate word *c* in the ranked lists for all seed words are then summed:

$$SeedBM25_c = \sum_{s=1}^{S} QACW_{c,s} \qquad (3)$$

The SeedBM25 scores for the candidate words are then normalised to be in the range between zero and one, so that they can be used in computing the MWU score according to the method described in Section 2.3.2.

2.3 Extracting and ranking multiword units

Here we describe a method for extracting MWU aspects and ranking them by the likelihood of belonging to the target category of review aspects. This stage is represented as Box 2 in Figure 1.

2.3.1 Identifying MWU aspects

Aspects of user-written reviews are frequently multiword units, many of which have a rather complex syntactic structure, including prepositions and conjunctions. Examples of syntactically complex dish names labeled by annotators in our experiments include: "pasta with lamb, olives, goat cheese and rosemary", "coconut ice cream with sticky rice". Currently available state-of-the-art Noun phrase (NP) chunkers, for example, Illinois chunker (Punyakanok and Roth, 2001), do not identify complex syntactic structures, such as the above. For instance, for the last phrase above, the Illinois chunker outputs two separate NPs: "coconut ice cream" and "sticky rice". Another problem in extracting complex MWU aspect names is identifying their proper boundaries. For instance, the phrase "good food and friendly service" contains two aspects ("good food" and "friendly service). Lastly, many aspects in user-written reviews contain subjective modifiers, e.g., "delicious seafood pasta", which must be identified and removed.

In this section we present a novel algorithm that "builds" MWU aspects in a bottom-up manner. We begin the process with the single nouns, which are the governing words in syntactic dependency relations, and by following a set of syntactic rules, we merge them with the adjacent words in an iterative manner, one at a time. In addition to syntactic rules, we also calculate the degree of co-occurrence association using Normalized Pointwise Mutual Information (NPMI) between two strings to be merged. Only if all the syntactic rules are satisfied and the NPMI is above the threshold, then the two strings are merged. The algorithm is presented as a pseudocode in Table 1, and described in more detail below.

```
1   For each document
2       Do dependency parse of the document
3       For each sentence
4           #Merge Stage 1: merge contiguous "nn", "amod" and "poss" relations
5           Until no more merges
6               For each governing word g in a dependency relation
7                   If POS (g) = "NN"
8                       For each word d depending on g
9                           Get all words (inBetweenWords) in text between g and d
10                          If inBetweenWords = 0
11                              Do CheckAssociationMergeStage1
12                              If CheckAssociation=passed
13                                  Merge d with g
14          #Merge Stage 2: merge contiguous "prep_*" and "conj_and" relations
15          Until no more merges
16              For each governing word g in a dependency relation
17                  If POS(g) = "NN"
18                      For each word d depending on g
19                          Get all words (inBetweenWords) in text between g and d
```

```
20                  If inBetweenWords consist only of (i) the same word as the name of
21              the dependency relation (e.g. "and" in "conj_and"), (ii) a token with POS="DT", (iii)
22              a token with POS=","
23                      Do CheckAssociationMergeStage2
24                      If CheckAssociation=passed
25                          Merge d with inBetweenWords and g

26  CheckAssociationMergeStage1
27      If POS(d) = "NN"
28              If relation is "nn" or "amod"
29                  CheckAssociation=passed
30              Else If relation is "poss"
31                  Calculate NPMI
32                  If NPMI > threshold_stage1
33                      CheckAssociation=passed
34      Else
35          If relation is "nn" or "amod"
36                  Calculate NPMI
37                  If NPMI > threshold_stage1
38                      CheckAssociation=passed

39  CheckAssociationMergeStage2
40      If relation is "prep_in" or "prep_of" or "prep_over" or "prep_under" or "prep_with" or "conj_and"
41          If POS(d) = "NN"
42                  Calculate NPMI
43                  If NPMI > threshold_stage2
44                      CheckAssociation=passed
```

Table 1. Algorithm for identifying MWU aspects.

In Stage 1 of the algorithm (lines 4-13 in Table 1), we iteratively merge words that have one of the following dependency relations: noun compound modifier (*nn*), adjectival modifier (*amod*), and possession modifier (*poss*). Examples of each respective relation are: "field salad" (*nn*), "balsamic vinegar" (*amod*), "chef's choice" (*poss*). We split the document into sentences and perform a dependency parse of each sentence using the Stanford dependency parser. For each dependency relation in a sentence, e.g., "*amod*(vinegar, balsamic)", we take its governing word *g* (here "vinegar"), and check if it is a noun (line 7 in Table 1). If it is, we then take all other dependency relations in the same sentence where it is also a governing word and from each we take the word *d* that depends on it, provided that it is immediately next to *g* in text (lines 9-10) We then apply a set of rules to check if *d* and *g* should be merged into one MWU (module *CheckAssociationMergeStage1*, lines 26-38). Specifically, if *d* is a noun, and the relation is *nn* or *amod* we merge them. In the case when the relation is *poss*, or when *d* is not a noun and relation is either *nn* or *amod*, we calculate Normalized Pointwise Mutual Information (NPMI) between *d* and *g*. Only if NPMI is above the predefined threshold (*threshold_stage1*), the two words are merged. The calculation of NPMI is described in Section 2.3.1.1. Merging in our algorithm means that the following actions are taken: *g* is modified by appending *d* to it in the correct order, *d* is deleted, the dependency relation between *d* and *g* is deleted. The algorithm then continues to the next iteration with the updated data structures. The process stops when no more merges take place. Note that the POS of *g* does not change throughout the iterations (e.g. if we start with "vinegar" as *g* (POS="NN") and then add "balsamic" as *d* to it, the POS of the new *g*, "balsamic vinegar", stays as "NN").

After all the valid merges have been made in Stage 1, the algorithm proceeds to Stage 2 (lines 14-25), where we merge words that have either a valid prepositional relation *prep_**, i.e. connected with a preposition "in", "of", "over", "under" or "with", or have a *conj_and* relation, which means that the two words are connected by coordinating conjunctions "and" or ",". Similar to Stage 1, we take each governing word *g*, and from each dependency relation containing *g* we take its depending word *d*. For each *d*, we take all the words in text between *g* and *d* (referred to as *inBetweenWords*). We then check that *inBetweenWords* consist only of the following: one word that matches the name of the relation, (e.g., if the relation is *prep_in*, then the valid word would be "in"), one optional determiner, such as "the" (i.e., token with POS="DT"), and one optional comma (i.e., token with POS=","). We also ensure that *d* is a noun, that there exists one of the above valid relations between *g* and *d*, and that the NPMI between *g* and *d* is above the predefined threshold (*threshold_stage2*). If all the above conditions are satisfied, we modify *g* by appending to it *d* and *inBetweenWords* in the correct order. For example, in the sentence "I had fish and chips.", one of the dependency relations is *conj_and*(fish-3, chips-5), with "fish" as *g*, "chips" as *d* and "and" as *inBetweenWords*. After merging of all three in the correct order of occurrence, we get MWU "fish and chips". After each merge, *g* is updated, *d* and every token in *inBetweenWords* are deleted, and the dependency relation between *d* and *g* is deleted. The process continues to the next iteration with the updated data structures, and stops when no more merges are made.

Below is a walk through the merging process based on an example sentence: "Everything was great from the tasty braised short ribs with creamy saffron risotto to the delicious dessert." Figure 2 shows the POS tags, while Table 2 shows the grammatical dependency triples. Both POS tagging and dependency parsing were performed using the Stanford CoreNLP package.

```
Everything/NN was/VBD great/JJ from/IN the/DT tasty/JJ braised/JJ short/JJ ribs/NNS
with/IN creamy/JJ saffron/NN risotto/NN to/TO the/DT delicious/JJ dessert/NN ./.
```

Figure 2. POS tagged sentence.

Relation name	**Governing word *g* (position)**	**Depending word *d* (position)**	**Relation description**
nsubj	great (2)	Everything (0)	nominal subject
cop	great (2)	was (1)	copula
det	ribs (8)	the (4)	determiner
amod	braised (6)	tasty (5)	adjectival modifier
amod	ribs (8)	braised (6)	adjectival modifier
amod	ribs (8)	short (7)	adjectival modifier
prep_from	great (2)	ribs (8)	preposition "from"
amod	risotto (12)	creamy (10)	adjectival modifier
nn	risotto (12)	saffron (11)	noun modifier
prep_with	ribs (8)	risotto (12)	preposition "with"
det	dessert (16)	the (14)	determiner
amod	dessert (16)	delicious (15)	adjectival modifier
prep_to	great (2)	dessert (16)	preposition "to"

Table 2. Dependency triples.

Table 3 illustrates the merging process that takes place in Stage 1. In the first iteration, "short" and "ribs" are merged into "short ribs", while "saffron" and "risotto" are merged into "saffron risotto". "Delicious" and "dessert" are not merged because their NPMI is below the *threshold_stage1* value (0.5). In the second iteration, we merge "creamy" with "saffron risotto", and “braised” with “short ribs”. Italicised words in the last column represent intermediate merge results, i.e. those that were merged into larger phrases in the subsequent iteration.

Iteration	***d*** **(position)**	***g*** **(position)**	**Relation**	**NPMI**	**Decision**	**Updated *g***
1	short (7)	ribs (8)	amod	0.5066	merge	*short ribs*
1	delicious (15)	dessert (16)	amod	0.0867	no merge	dessert
1	saffron (11)	risotto (12)	nn	no test	merge	*saffron risotto*
2	creamy (10)	saffron risotto (12)	amod	0.5015	merge	creamy saffron risotto
2	braised (6)	short ribs (8)	amod	0.6522	merge	braised short ribs

Table 3. The merge process in Stage 1.

In Stage 2 (Table 4), we merge “creamy saffron risotto” with “braised short ribs” as they satisfy all the rules, including the NPMI, which is above *threshold_stage2* value (0.75).

d **(position)**	***g*** **(position)**	**Relation**	**inBetween Words**	**NPMI**	**Decision**	**Updated *g***
creamy saffron risotto (12)	braised short ribs (8)	prep_with	with	0.9156	merge	braised short ribs with creamy saffron risotto

Table 4. The merge process in Stage 2.

2.3.1.1 NPMI

Pointwise Mutual Information (Church and Hanks, 1990) is one of the most commonly used word co-occurrence association measures. One characteristic of PMI is that it has no upper bound, which doesn’t allow us to compare PMI values against a fixed threshold in our task. The other well-known PMI characteristic is that it favours low-frequency words. Instead, we use Normalised PMI (NPMI), which was proposed by Bouma (2009). NPMI (Eq. 4) has fixed upper and lower bounds. When two words always occur together NPMI is 1; when they occur by chance, it is close to 0, when they always occur separately, it equals -1. NPMI also has been shown to have lower sensitivity to low-frequency words than PMI (Bouma, 2009).

$$NPMI(x,y) = \left(\log\frac{p(x,y)}{p(x)p(y)}\right) / -\log p(x,y) \qquad (4)$$

Where: $p(x,y)$ is calculated as $f(x,y)/N$, where $f(x,y)$ is the number of times y occurs immediately following x in the corpus, and N is the number of word occurrences (tokens) in the corpus; $p(x) = f(x)/N$; $p(y) = f(y)/N$.

We used the corpus of 157,865 restaurant reviews (*N*=13,712,600) to calculate NPMI. When we calculate co-occurrences between phrases that have words occurring between them (*inBetweenWords*), we append the *inBetweenWords* to the first phrase in the correct order of occurrence. For instance to calculate NPMI between the phrases in Table 4, we set *x* to “braised short ribs with”, and *y* to “creamy saffron risotto”.

We evaluated the following values of NPMI thresholds for Stage1 (*threshold_Stage1*) and Stage2 (*threshold_Stage2*): 0.25, 0.5 and 0.75. The best results were obtained with *threshold_Stage1*=0.5 and *threshold_Stage2*=0.75. The results for all combinations of values are presented in Section 3.

2.3.1.2 Weighted NPMI

One of the common types of merges that take place in Stage 1 are “JJ + NN”, i.e. between an adjective and a noun. The use of NPMI in determining whether to merge an adjective with a noun does not distinguish between subjective vs. non-subjective adjectives, such as “scrumptious” vs.

"spicy". While subjective adjectives that occur with a large number of different nouns are often not merged (e.g. "delicious" in Table 3), some of the lower frequency subjective adjectives can have high NPMI scores with their co-occurring nouns. To rectify this problem, we propose a weighted NPMI approach, whereby we combine the NPMI score of the "JJ + NN" pair with the subjectivity score of the adjective. The subjectivity scores (*Subj*) of all adjectives in the corpus are pre-computed (Box 1a in Fig.1) by using the same semi-supervised method that is used for calculating similarity of nouns to seed dish names detailed in Section 2.2. Here, we use a small set of subjective adjectives as seeds and all adjectives extracted from the corpus of restaurant reviews as candidate words. The subjectivity scores of all adjectives in the corpus are normalised to be between 0 and 1. The NPMI is then combined with the subjectivity score of the adjective using Eq. 5.

$$weightedNPMI(a,n) = (NPMI(a,n)\times\alpha) + \Big(\big(1 - Subj(a)\big)\times(1-\alpha)\Big) \qquad (5)$$

Where: a – adjective, n – noun, α – tuning constant.

The following values of α were evaluated: 0.3, 0.5 and 0.7. The results of the evaluation are detailed in Section 3.

2.3.2 Weighting MWU aspects

The final goal in our task is to find in each review the MWU aspects that belong to the desired category (dish names, in our example). For this we need to calculate a weight for each MWU aspect, representing its likelihood of belonging to this category, and rank them by this weight. Our approach to weighting MWU aspects is to combine the pre-computed similarity scores of single nouns they contain to seeds, as described in Section 2.2.

The process of computing MWU weights is done in two phases:

Phase 1. Here we compute the weights of all MWUs output by Stage 1 of the MWU merging process (see Table 1). The weights are computed immediately after the completion of all Stage 1 merges, and only MWUs with weight>0 are considered for merging in Stage 2.

The MWUs output by Stage 1 are simple NPs, that is, consisting of zero or more adjectives followed by one or more nouns. The rightmost noun in the NP is treated as the head of the NP. We hypothesise that the further away the noun is from the head of the NP, the less its score should contribute to the overall score of the NP. For instance, if we have a seed-similarity score of 0.5 for "pizza" and 0.2 for "restaurant", intuitively "restaurant pizza" should be weighted higher than "pizza restaurant". To achieve this we discount noun scores based on the distance from the end of the NP. We evaluate four discount functions (Table 5).

Log-linear	$D = 1 - log_{10}(d_i)$
Linear	$D = 1 - ((d_i - 1)\times 0.1)$
No discount	$D = 1$
0.5 discount	$D = \begin{cases} 1 & if\ d_i = 1 \\ 0.5 & otherwise \end{cases}$

Table 5. Discount factors.

Here d_i is the distance of the noun i from the end of the NP, with d of the last word being 1.

$$NPscore = \frac{\sum_i^n Dw_i}{n} \qquad (6)$$

Where: w_i – seed-similarity score of the noun i calculated according to the method presented in Section 2.2.2 or using any of the other three similarity measures that were evaluated in our experiments; D – discount factor, n – number of nouns in the NP.

In Section 3.1, we report performance comparison of these discount functions. "No discount" gives the best results, therefore it was used throughout the experiments reported in Section 3.

In Phase 1 we also assign the weight of zero to those NPs, whose head noun is below the cut-off threshold in the list of single nouns ranked by seed-similarity. The cut-off threshold is set empirically. In our experiments, we set it to 1000 words. NPs with the weight of zero are not considered for merging in Stage 2 of the merging process.

Phase 2. In this phase, we compute the weights of MWUs output by Stage 2 of the MWU merging process. Stage 2 outputs complex MWUs, which consist of two or more NPs. The weight of an MWU output by Stage 2 is the average of the weights of its constituent NPs (calculated in Phase 1).

3. Evaluation

The goal of the evaluation is to determine the effectiveness of the proposed method for extracting and ranking multiword review aspects belonging to the target aspect category. Restaurant dish names are used as the target aspect category in our experiments. We consider this to be one of the most challenging aspect categories to extract as there are many syntactically complex phrases among dish names, many of which are either unique or low-frequency.

The dataset used for evaluation consists of 157,865 restaurant reviews from one of the major business review websites, provided to us by a partner organisation. The collection contains reviews for 32,782 restaurants in the United States. The average number of words per review is 64.7. We randomly selected 600 reviews, where all dish names and their subjective modifiers were manually labeled by two annotators. Each annotator labeled 300 reviews, then the third annotator acted as adjudicator, going through all labeled dish names and correcting errors, i.e. non-dish names labeled as dish names by mistake. In total, 1000 multiword and single-noun distinct dish names were labeled by the annotators in 600 reviews. The annotators labeled different sets of reviews, therefore inter-annotator agreement cannot be calculated. However, prior to this, the two annotators were asked to label dish names in the same set of 50 reviews. The annotators were not labeling a fixed set of items, which means that they may disagree on the boundaries of an expression, or the presence/absence of an annotation. For instance, one annotator may label "braised short ribs with creamy saffron risotto" as one dish name, whereas another could label "braised short ribs" and "creamy saffron risotto" as two dishes. For these reasons, it is not possible to use Kappa statistic to calculate inter-annotator agreement. Wiebe, Wilson and Cardie (2005) pointed out these annotation problems when there is no fixed set of items to label, and suggested the *agr* metric (Eq. 7).

$$agr(a \parallel b) = \frac{|A\ matching\ B|}{|A|} \qquad (7)$$

Where: A is a set of text strings marked by the annotator *a* as dish names; and B are text strings marked by annotator *b* as dish names. Annotator *a* labeled 156 dish names, and annotator *b* labeled 143 dish names. They agreed on 105 dish names. Table 6 shows the inter-annotator agreement.

$agr(a\|\|b)$	$agr(b\|\|a)$	Average
0.67	0.73	0.7

Table 6. Inter-annotator agreement.

As can be see from this table, the annotators can identify dish names with reasonable accuracy.

We generated 20 seed sets, each consisting of 10 seed words. Sets 1-10 were used for tuning the system parameters, while sets 11-20 were used for testing. Seed sets were generated as follows: all single nouns were extracted from the 1000 dish names, from which the annotators selected only those nouns that refer to food. The reason why we did not automatically use all nouns is that some nouns in MWU dish names, do not refer to food, for example, "field" in "field salad". In this manner, 573 unique single-word food/dish names were identified. They were then ranked by frequency of occurrence in the 600 reviews, and 20 seed sets were generated randomly from the list of 100 top-ranked nouns. As an example, one of the seed set consists of the following words: "roast", "salad", "potatoes", "shrimp", "roll", "lime", "mushrooms", "sushi", "cake", "wine".

Evaluation was done per review, as we are interested in knowing how well each method ranks MWUs extracted from each review by the likelihood of being dish names. The output of the method is a ranked list of dish names per review. For each review Average Precision (AveP) is calculated, which is then averaged across all 600 reviews, giving Mean Average Precision (MAP). AveP is calculated in the same way as in document retrieval, except that here instead of having a ranked list of documents retrieved for a topic, we have a ranked list of dish names retrieved from each restaurant review. To sum up, to calculate AveP for each review we calculate precision at each dish name retrieved; these precision values are then averaged, dividing the sum of precision values by the total number of dish names for that review.

As seed-similarity measures, we evaluated BM25-based method (Section 2.2), Lin's method (Section 4.1.1), Weeds' method (Section 4.1.2) and BalAPinc (Section 4.1.3).

The following three MWU extraction methods were evaluated with each seed-similarity measure:

- *NP* (baseline) – this method merges all words that conform to the "JJ(≥0) NN(≥1)" pattern, i.e. zero or more adjectives followed by one or more nouns.
- *NPMI* – uses the two-stage MWU merging process as outlined in Section 2.3.1, with the Normalised Pointwise Mutual Information (Section 2.3.1.1) as the association measure.
- *weightedNPMI* – based on the same two-stage process, but using weighted NPMI method described in Section 2.3.1.2.

The MWU ranking in all three MWU extraction methods is done using the same approach, outlined in Section 2.3.2, with the "no discount" function, which showed slightly better performance than other discount functions (see Section 3.1).

The best results in both training and test seed sets with all four seed-similarity measures were obtained with *stage1_threshold*=0.5 and *stage2_threshold*=0.75 in the MWU merging process. Table 6 shows the results of different MWU extraction methods with different seed-similarity measures. The results of NPMI and weightedNPMI methods are compared to NP as the baseline (used with the same seed-similarity function). Statistical significance test was done using paired t-test. Results showing significant improvements over the corresponding baseline NP run are indicated with "a" superscript, while significant improvements of weightedNPMI runs over the corresponding NPMI runs are indicated with "b" superscript. Both are at 0.01 significance level.

As can be seen from Table 7, the MWU merging process with NPMI showed statistically significant improvements over the baseline NP runs with all seed similarity measures. The improvements gained are at least 17%. The combination of NPMI with the subjectivity score of adjectives (weightedNPMI runs) showed statistically significant improvements over NPMI in some runs.

Method	**Training seed sets**			**Test seed sets**		
	MAP	**>NP (%)**	**>NPMI (%)**	**MAP**	**>NP (%)**	**>NPMI (%)**
BM25						
NP-BM25 baseline	0.4945			0.4806		
NPMI-BM25	0.5786[a]	17		0.5637[a]	17.3	
weightedNPMI-BM25 (α=0.3)	0.5886[ab]	19	1.7	0.5747[ab]	19.6	1.9
Lin						
NP-Lin baseline	0.5008			0.5009		
NPMI-Lin	0.5900[a]	17.8		0.5897[a]	17.7	
weightedNPMI-Lin (α=0.3)	0.5950[ab]	18.8	0.8	0.5971[ab]	19.2	1.2
Weeds						
NP-Weeds baseline	0.4697			0.4730		
NPMI-Weeds	0.5569[a]	18.6		0.5593[a]	18.2	
weightedNPMI-Weeds (α=0.5)	0.5602[a]	19.3	0.6	0.5659[ab]	19.6	1.2
balAPinc						
NP-balAPinc baseline	0.4946			0.4937		
NPMI-balAPinc	0.5810[a]	17.5		0.5787[a]	17.2	
weightedNPMI-balAPinc (α=0.5)	0.5821[a]	17.7	0.2	0.5829[ab]	18.1	0.7

Table 7. Results of MWU extraction methods with different seed-similarity functions.

In order to understand how much seed-similarity measures contribute to performance of the overall method, we conducted a run (IDF-NP), which merges all words that conform to the "JJ(≥0) NN(≥1)" pattern (the same rule as in NP runs in Table 7), but instead of a seed-similarity measure, IDF-NP uses the average of IDF values of all nouns in the MWU as its weight, which is then used for the ranking of MWUs. The MAP of IDF-NP is 0.2129, which is significantly lower than the MAP of all NP runs that use seed-similarity measures.

Table 8 contains the results of NPMI-BM25 method with different values of *stage1_threshold* and *stage2_threshold* on the test seed sets.

stage2 threshold / *stage1 threshold*	0.25	0.5	0.75
0.25	0.4988	0.5442	0.5578
0.5	0.5049	0.5499	**0.5637**
0.75	0.4927	0.5347	0.5503

Table 8. Results (MAP) of NPMI-BM25 with different *stage1_threshold* and *stage2_threshold* values on test seed sets.

3.1 Discount factors in NPscore

In Section 2.3.2, we hypothesised that discounting scores of single nouns the further away they are from the head of the noun phrase is useful. We proposed four discount factors: "0.5 discount", "log-linear", "linear", "no discount" (Table 5). The experiments did not support our hypothesis, showing that "no discount" gives the best results, although "linear" gave a similar, but slightly worse performance. Figure 3 shows the comparison of the discount factors applied to NPMI runs on 10 test seed sets.

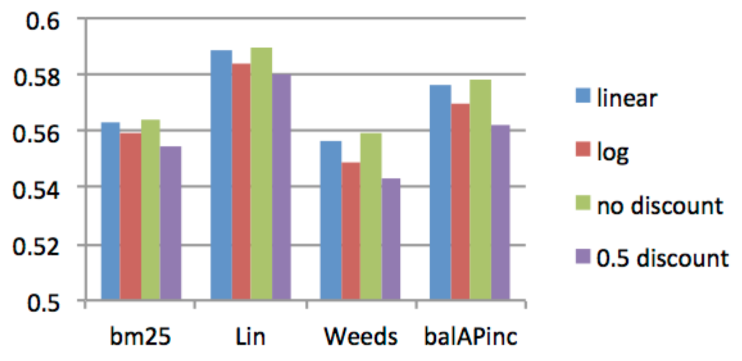

Figure 3. MAP of NPMI runs with different discount factors.

Analysis of the effect of "linear" and "no discount" functions on performance for one of the test seed sets with the BM25 similarity function shows that out of 536 documents that have at least one dish name, 37 had a higher Average Precision (AveP) with "no discount", with the average improvement of 8%, and 24 had a higher AveP with "linear" discount, with the average improvement of 21%.

3.2 Regression analysis

The NPMI method identifies the candidate strings for merging based on syntactic rules, and then uses only the strength of their co-occurrence association in deciding whether to merge them or not. In addition to the co-occurrence association strength between text strings, there are a number of other features available to us that could be useful in making the correct merge decision. Specifically, we have the following features: length of the strings that we are trying to merge, their part of speech and the dependency relationship between them. We performed a binary logistic regression analysis with the goal of learning a regression function that can be used in deciding whether or not to merge two strings.

The regression analysis was done using 10-fold cross validation. First, we split the 600 documents randomly into 10 sets. For each relevant MWU in each document, we generate tuples representing all possible contiguous combinations of n-grams. For example, if we have a relevant MWU "garlic green beans", we generate four possible combinations of n-grams from it:

- *garlic, green*
- *garlic, green beans*
- *garlic green, beans*
- *green, beans*

We will refer to this set of possible n-gram combinations as a relevant set {R}. Next, we run our MWU merging algorithm, as described in Table 1, however, instead of doing NPMI test, we check whether the combination of strings to be merged is in the relevant set {R} or not. If it is, we merge it, and add it as a relevant case into the dataset that will be used for regression. If it is not in {R}, we do not merge it, and add it as a non-relevant set into the dataset to be used for regression. For each relevant and non-relevant case the following features are recorded:

S1-length – number of words in the first string (S1). The first string is the string that appears before the second string in text;
S2-length – number of words in the second string (S2);
Relation – dependency relation between S1 and S2;
NPMI – Normalised Pointwise Mutual Information between S1 and S2;
Relevance – 1 – relevant (correct) merge (i.e. the n-gram combination is in the relevant set {R}), 0 – non-relevant;
Set – 1-10 (used for 10-fold cross-validation).

The "Relation" is a categorical variable that can have the following values: amod, conj_and, nn, poss, prep_in, prep_with, prep_other. We grouped all prepositional relations that are neither "with", nor "in" into "prep_other", since they are very sparse.

We experimented with using part of speech as a feature, but it did not have any effect on performance, since there is some degree of redundancy between part of speech and dependency relation, e.g. "amod" in our dataset is almost always a relationship between adjective and noun, whereas "nn" is always between two nouns.

Next, we performed 10 logistic regression iterations (folds) using the 10-fold cross-validation method. For each fold, the training was done on 9 sets, and tested on the remaining set. For instance for fold 1, the training was done on sets 2-10, while testing on set 1. We rotated the sets used for training and testing. The dependent variable is "Relevance", while "S1-length", "S2-length", "Relation" and "NPMI" are independent variables. Table 9 shows the regression coefficients for the variables in regression for each of the 10 folds. The superscript "a" indicates that the variable is significant at 0.05 significance level, while "b" indicates significance at 0.01 significance level. The entire categorical variable "Relation" is significant across all folds, as well as its individual parameters of "conj_and" and "prep_other". NPMI is also significant across all folds. Both the high regression coefficient for NPMI and its significance indicate that NPMI contributes to the model with the higher average NPMI for the correct (relevant) merges. S1-length contributes to the model in all but two folds. The negative regression coefficient indicates that correct merges tend to have shorter first string than the incorrect merges. The length of the second string does not contribute to the model. The negative regression coefficient for "conj_and" and "prep_other" indicate that these relations are present in a larger number of incorrect merges than correct.

Folds Feature	1	2	3	4	5	6	7	8	9	10
S1-length	-.444[a]	-.506[a]	-.559[a]	-.419	-.601[a]	-.491[a]	-.513[a]	-.474[a]	-.548[a]	-.432
S2-length	.054	.093	.037	.013	.086	.100	.161	.074	.127	-.045
Relation[b]										
Relation (amod)	.031	.029	-.220	.301	-.155	-.080	-.220	-.039	-.242	-.019
Relation (conj_and)	-1.721[b]	-1.804[b]	-2.078[b]	-1.681[b]	-1.888[b]	-1.997[b]	-1.991[b]	-1.998[b]	-2.077[b]	-2.001[b]
Relation (nn)	.861	.841	.607	1.145[a]	.671	.773	.619	.786	.539	.681
Relation (poss)	-1.004	-.517	-1.474	-.888	-1.502	-1.578	-1.513	-1.404	-1.409	-1.251
Relation (prep_in)	-.934	-.859	-1.146	-.820	-.598	-.642	-1.391[a]	-.915	-1.138	-1.048
Relation (prep_other)	-1.759[b]	-1.763[b]	-1.898[b]	-1.744[b]	-1.844[b]	-2.137[b]	-1.946[b]	-1.837[b]	-2.145[b]	-2.453[b]
NPMI	4.780[b]	4.898[b]	4.947[b]	5.006[b]	4.977[b]	4.806[b]	4.773[b]	4.847[b]	4.900[b]	4.862[b]
Constant	-1.742[a]	-1.719[a]	-1.382	-2.073[b]	-1.528[a]	-1.654[a]	-1.551[a]	-1.651[a]	-1.436	-1.551[a]

Table 9. Regression coefficients for the variables used in the regression analysis.

The following function was then used to calculate the probability of the correct merge:

$$p = \frac{e^{(\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_i x_i)}}{1+e^{(\alpha+\beta_1 x_1+\beta_2 x_2+\cdots+\beta_i x_i)}} \qquad (8)$$

Where α – constant, β_i – regression coefficient, x_i – variable value. Two strings were merged if $p > 0.5$.

We performed 10 MWU extraction runs, one for each fold, with each distributional similarity measure. For example, in the first run, the coefficients for the first fold were used, with documents in set 1 used for evaluation. The MAP value was calculated for each of the 10 folds, and average was computed. The results are presented in Table 10. For comparison, the NPMI results are repeated from Table 7. Runs marked with "*" have statistically significant difference at 0.01 level, compared to NPMI runs.

Method	**Training seed sets**		**Test seed sets**	
	MAP	**>NPMI (%)**	**MAP**	**>NPMI (%)**
BM25				
NPMI-BM25	0.5786		0.5637	
regression-BM25	0.5918*	2.27	0.5767*	2.31
Lin				
NPMI-Lin	0.5900		0.5897	
regression-Lin	0.6040*	2.36	0.6042*	2.46
Weeds				
NPMI-Weeds	0.5569		0.5593	
regression-Weeds	0.5684*	2.05	0.5716*	2.19
balAPinc				
NPMI-balAPinc	0.5810		0.5787	
regression-balAPinc	0.5937*	2.18	0.5924*	2.37

Table 10. Results of the MWU extraction methods using regression function.

The results presented in Table 10 show that there is a statistically significant improvement in using the regression function compared to just NPMI.

4. Related work

Previous work on aspect extraction from user reviews did not focus on both identifying compound aspect names and ranking them by the likelihood of belonging to a specific aspect category. For instance, Yi et al (2003), Hu and Liu (2005), Popescu and Etzioni (2005), Jo and Oh (2011) proposed methods for extracting aspects in general, without grouping them into categories. Yi et al (2003) extracted noun phrases conforming to the pattern "JJ(0-2) NN(1-3)". They also experimented with extracting only those noun phrases that are preceded by a definite article "the", and those that occur at the beginning of the sentence and are followed by a verb phrase. Blair-Goldensohn et al. (2008) used a supervised method to identify complete sentences that refer to a coarse-grained aspect category. As part of their method, they extracted frequent aspects that are nouns or noun compounds of up to three words, occurring in either opinionated sentences or in certain syntactic patterns, such as following an adjective. A number of unsupervised approaches have also been proposed that automatically identify categories of aspects in the corpus of reviews and single words that are statistically associated with these categories (Titov and McDonald, 2008; Brody and Elhadad, 2010)

The novelty of our method compared to the previous works on aspect extraction is that it identifies complex aspect names that belong to a specific category of aspects that can be defined by the end-user or application developer as a small set of seed words.

There exist a number of corpus-based methods for extracting words belonging to the same semantic category. Based on the amount of training data required, there are three categories of methods: unsupervised, semi-supervised and supervised. Tsai and Chou (2011) proposed an unsupervised method for the extraction of dish names from Chinese blogs using suffix arrays and conditional random fields. Supervised methods typically use classifiers trained on a large amount of manually annotated data, e.g., (Rahman and Ng, 2010). Semi-supervised methods require only a small amount of training data, usually in the form of seed words. These methods are generally more attractive for practical applications than supervised methods since less manual labour is needed. Examples of semi-supervised methods are Meta-Bootstrapping (Riloff and Jones, 1999), Basilisk (Thelen and Riloff, 2002) and Snowball (Agichtein and Gravano, 2000). Semi-supervised approaches rely on either (a) a number of hand-crafted extraction patterns, or (b) co-occurrence information of lexical units, or (c) their distributional similarity. A comprehensive review of co-occurrence-based and distributional similarity approaches is given in (Weeds and Weir, 2006).

One of the earliest methods using extraction patterns is a hyponymy detection method by Hearst (1992), who uses six patterns (e.g. "such X as *") and a category name X to identify its hyponyms. Lately, a number of other methods using patterns were proposed, e.g. (Wang and Cohen, 2009; Etzioni et al., 2005; Kozareva, Riloff and Hovy, 2008). Wang and Cohen (2009) apply Hearst's hyponymy patterns to get an initial set, and then expand it by a web-based algorithm from the pages containing seeds. Etzioni et al. (2005) also use a set of hyponymy patterns, and select candidates using Pointwise Mutual Information (PMI). Kozareva et al. (2008) use a doubly-anchored pattern "X such as Y and *" and a bootstrapping algorithm. All these methods require the scale and redundancy of the web, as the extraction patterns may not be frequently observed in smaller corpora.

Co-occurrence based methods rely on such statistical measures as PMI (Church et al., 1991), $\chi2$ (Chi-square) (Manning and Schütze, 1999) and Log-likelihood ratio (Dunning, 1993). Riloff and Shepherd (1997) rely on the co-occurrence of nouns in small window sizes with a small set of seeds. Yarowsky (1992) uses co-occurrence of words in windows of 100 words with words from a specific Roget thesaurus category to identify lists of words salient to this category. One limitation of the co-occurrence based measures is that words have to occur in the vicinity of the known words in order to be extracted. This limitation is overcome by distributional similarity methods.

According to the distributional similarity principle, words that occur in similar contexts are likely to have similar meanings. A number of symmetric and asymmetric distributional similarity measures have been proposed. Measures proposed by Lin (1998) and Weeds and Weir (2006) are the most well-known symmetric distributional similarity measures. Lin's measure uses grammatical dependency relations as features, weighted using Mutual Information. Weeds and Weir propose a general framework for computing distributional similarity measure based on the concepts of precision and recall. Kotlerman et al. (2009) propose an asymmetric (directional) similarity measure, balAPinc, designed to find words with more specialised meaning compared to the seed. They adapt the concept of average precision from Information Retrieval to calculate similarity between two words. In their approach the features in the vector of the seed word are analogous to the complete set of relevant documents, while the features in the vector of the candidate word are analogous to the retrieved documents.

Distributional similarity methods also differ by the linguistic units they use as context. For example, Pantel et al. (2009) use noun phrase chunks to the left and right of a term as its context, while Lin (1998) and Kotlerman et al. (2009) use grammatical dependency relations as features. Weeds and Weir (2006) use verbs as features when calculating similarity between nouns. As discussed in (Kilgarriff and Yallop, 2000), the use of grammatical relations as features leads to the identification of "tighter" relationships between words, whereas the use of document- and sentence-level word co-occurrences would lead to the identification of "looser" relationships. So, while the former are better for identifying words belonging to the same semantic class, the latter are more appropriate for grouping words into subject categories. In our approach we use the former.

4.1 Overview of distributional similarity measures used in this study

In this section we describe three distributional similarity measures that were evaluated as part of our method. Scores calculated using Lin's method (Eq. 10), Weeds' method (Eq. 15), and balAPinc (Eq. 19) for each candidate word with respect to each of the seeds are then normalised and added in the same way as in the BM25 method (Eq. 3) to obtain one score for each candidate entity.

4.1.1 Lin's measure

Lin's method (Lin, 1998) uses grammatical dependency relations as features, weighted using Mutual Information. Each feature is a dependency triple (*w, r, w'*), where *w* and *w'* are words and *r* is a grammatical relation. Each feature is weighted by using Mutual Information as follows:

$$I(w,r,w') = log\frac{f(w,r,w')\times f(*,r,*)}{f(w,r,*)\times f(*,r,w')} \tag{9}$$

Where: *f(w, r, w')* – the frequency of occurrence of the triple *(w, r, w')*, * – wildcard, for example, *f(*, r, w')* means the sum of frequencies of all dependency triples matching the pattern where the grammatical relation is r and the second word is *w'*. For example, if the entity is "pasta" and the feature is “eat VB:dobj:NN pasta”, the Mutual Information between “pasta” and the feature will be calculated as follows:

$$I(eat, VB{:}dobj{:}NN, pasta) = log\frac{f(eat, VB{:}dobj{:}NN, pasta)\times f(*, VB{:}dobj{:}NN, *)}{f(eat, VB{:}dobj{:}NN, *)\times f(*, VB{:}dobj{:}NN, pasta)}$$

Each word (candidate or seed) is represented as a vector *T*, consisting of pairs *(r, w)* with $I > 0$. Similarity between a seed word s and a candidate word c is calculated as follows:

$$LIN(c, s) = \frac{\sum_{(r,w)\in T(c)\cap T(s)}(I(c, r, w) + I(s, r, w))}{\sum_{(r,w)\in T(c)} I(c, r, w) + \sum_{(r,w)\in T(s)} I(s, r, w)} \quad (10)$$

For calculating Mutual Information scores, we used dependency triple frequencies obtained from the entire corpus of 157,865 restaurant reviews. For this, we performed dependency parsing using Stanford parser, resulting in the total of 8,707,162 dependency triples (2,931,118 unique).

4.1.2 Weeds’ method

Weeds and Weir (2003) propose a general framework for computing distributional similarity measure based on the concepts of precision and recall. Precision is calculated as follows:

$$P(c, s) = \frac{\sum_{f\in T(c)\cap T(s)} D_c(f)}{\sum_{f\in T(c)} D_c(f)} \quad (11)$$

Where $D_c(f)$ is the degree of association between candidate word c and feature f. In their work, Weeds and Weir calculate similarity between nouns, using only verbs as features. They also discuss a number of possible word-feature association measures, including Mutual Information. In our work, as described earlier we used as features all dependency triples a word occurs in, and as a word-feature association measure we used Mutual Information (Eq 9).

Recall is calculated as follows:

$$R(c, s) = \frac{\sum_{f\in T(c)\cap T(s)} D_s(f)}{\sum_{f\in T(s)} D(f)} \quad (12)$$

Weeds and Weir propose to combine the two measures using harmonic mean, which optimises the product of Precision and Recall, and arithmetic mean, which optimises the sum of Precision and Recall. In calculating the arithmetic mean they use tuning constant β to control the relative contribution of Precision and Recall. They then propose to combine the two means, controlling their relative contribution by means of tuning constant γ.

$$m_h(c, s) = \left(\frac{2P(c, s)\times R(c, s)}{P(c, s) + R(c, s)}\right) \quad (13)$$

$$m_a(c, s) = \beta\times P(c, s) + (1 - \beta)\times R(c, s) \quad (14)$$

$$sim(c, s) = \gamma\times m_h(c, s) + (1 - \gamma)\times m_a(c, s) \quad (15)$$

In our experiments we evaluated on the 10 training dish name seed sets all possible combinations of β and γ, with values in the range from 0 to 1 in the increments of 0.1. The best performance was obtained with β=1 and γ=0.8.

4.1.3 BalAPinc

Kotlerman et al. (2009) propose a directional similarity measure, balAPinc, designed to find words with more specialised meaning compared to the seed. They adapt the concept of average precision from Information Retrieval to calculating similarity between two words. In their approach the features in the vector of the seed word are analogous to the complete set of relevant documents, while the features in the vector of the candidate word are analogous to the retrieved documents.

$$APinc(c \rightarrow s) = \frac{\sum_{r}^{|T_c|} P(r) \times rel(f_r)}{|T_c|} \tag{16}$$

Where f_r is the feature at rank r in the vector of the candidate word c. The features in each vector are ranked by Mutual Information according to Eq. 9.

$$rel(f) = \begin{cases} 1 - \frac{rank(f, T_s)}{|T_s| + 1} & \text{, if } f \in T_s \\ 0 & \text{, if } f \notin T_s \end{cases} \tag{17}$$

Where $rank(f,T_s)$ is the rank of feature f in the vector of the seed term s.

$$P(r) = \frac{|included\ features\ in\ ranks\ 1\ to\ r|}{r} \tag{18}$$

Where "included features in ranks 1 to r" are features that occur in both the seed word vector and in the ranked list of features in the candidate word vector between ranks 1 and r.

$$balAPinc(c \rightarrow s) = \sqrt{LIN(c,s) \times APinc(c,s)} \tag{19}$$

Where *LIN* is the Lin's similarity measure calculated according to Eq. 10.

5. Conclusion

In this paper we presented a method for identifying multiword aspects of user-written reviews that belong to a given category. The evaluation is done on the task of extracting dish names from restaurant reviews. The method initially computes distributional similarity between a seed word, representing a dish name, and each single noun in the corpus, and then produces a list of single nouns, ranked by similarity to all seeds. We compared three distributional similarity measures (Lin's, Weeds' and balAPinc), and a document retrieval function, BM25, adapted as a word similarity measure. In order to identify MWU boundaries, a combination of syntactic rules and a co-occurrence association measure was used. Specifically, we proposed a novel algorithm that "builds" MWU aspects in a bottom-up manner. The process begins with the single nouns that are governing words in syntactic dependency relations, and by following a set of syntactic rules, the adjacent words are merged with them in an iterative manner, one at a time. The decision of whether or not to merge two strings is determined by the syntactic rules, and the co-occurrence association between two strings, calculated using Normalized Pointwise Mutual Information.

The results indicate that the proposed syntactic rule-based method for identifying MWU boundaries combined with the co-occurrence association measure, NPMI, is useful, showing statistically significant improvement over the method that uses simple Noun Phrases of the form "JJ(≥0) NN(≥1)". In addition, the weighting of adjectives based on their similarity to subjective seed adjectives is also useful. We also performed a regression analysis, using variables, such as the length of word strings to be merged, the type of their dependency relationship, and NPMI. The results using the regression function showed statistically significant improvements compared to the method that only uses NPMI in deciding whether or not to merge two strings. Overall, regression analysis showed that three variables are useful in predicting correct and incorrect merges: NPMI, length of the first string and dependency relation. Specifically, regression analysis indicates that correct string merges tend to have higher NPMI and a shorter first string than incorrect merges. Also, statistically significant negative regression coefficients for “conj_and” (conjunctive “and” dependency relation) and “prep_other” (prepositional modifiers other than “with” and “in”) indicate that these relations are present in a larger number of incorrect merges than correct.

The proposed method is domain independent, as it uses only minimal supervision, i.e. a small set of seed words that can be provided by the end-user or application developer. Because the method does not rely on knowledge bases or annotated data sets, it is expected that it can be applied to other domains where entities tend to have syntactically complex names. In the future, we plan to evaluate the method on other domains.

References

Agichtein E. and Gravano L. (2000). Snowball: Extracting relations from large plain-text collections. In Proceedings of the 5th ACM Conference on Digital Libraries.

Blair-Goldensohn S., Hannan K., McDonald R., Neylon T., Reis G., Reynar J. (2008). Building a sentiment summarizer for local service reviews. In Proceedings of NLPIX 2008, Beijing, China.

Bouma G. (2009). Normalized (pointwise) mutual information in collocation extraction. In Proceedings of GSCL, pp. 31-40.

Brody S. and Elhadad N. (2010). An unsupervised aspect-sentiment model for online reviews. In Proceedings of HLT-NAACL’2010, pp. 804-812, Los Angeles, California.

Church K., Gale W., Hanks P., Hindle D. (1991). Using statistics in lexical analysis. In: Zernik U., ed. Lexical Acquisition: Using On-line Resources to Build a Lexicon. Englewood Cliffs, NJ, Lawrence Elbraum Associates, pp. 115-164.

de Marneffe M., MacCartney B. and Manning C. (2006). Generating Typed Dependency Parses from Phrase Structure Parses. In Proceedings of Language Resources and Evaluation Conference (LREC).

Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. Computational Linguistics, pp. 61-74.

Etzioni, O., Cafarella, M., Downey, D., Popescu A., Shaked, T., Soderland, S., Weld, D., Yates A. (2005). Unsupervised named-entity extraction from the web: an experimental study. Artificial Intelligence, 165(1), pp. 91-134.

Hearst M. (1992). Automatic acquisition of hyponyms from large text corpora. In Proceedings of the 14th Conference on Computational Linguistics, Nantes, France.

Hu M. and Liu B. (2004). Mining opinion features in customer reviews. In Proceedings of the National Conference on Artificial Intelligence (AAAI).

Jo Y. and Oh A. (2011). Aspect and sentiment unification model for online review analysis. In Proceedings of WSDM’11, February 9-11, 2011, Hong Kong, China.

Kilgarriff A. and Yallop C. (2000). What's in a thesaurus? In Proceedings of the Second International Conference on Language Resources and Evaluation, pp. 1371-1379.

Kotlerman L., Dagan I., Szpektor I., Zhitomirsky-Geffet M. (2009). Directional Distributional Similarity for Lexical Expansion. In Proceedings of ACL-IJCNLP, Singapore, pp. 69-72.

Kozareva Z., Riloff E. and Hovy E. (2008). Semantic Class Learning from the Web with Hyponym Pattern Linkage Graphs. In Proceedings of ACL-08: HLT, Columbus, USA.

Lin D. (1998). Automatic retrieval and clustering of similar words. In Proceedings of the 17th international Conference on Computational Linguistics, pp. 768-774.

Manning C. and Schütze H. (1999). Foundations of Statistical Natural Language Processing, MIT Press.

Pantel P., Crestan E., Borkovsky A., Popescu A. and Vyas V. (2009). Web-Scale Distributional Similarity and Entity Set Expansion. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 938-947. Singapore.

Popescu A. and Etzioni O. (2005). Extracting product features and opinions from reviews. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP).

Punyakanok V. and Roth D. (2001). The Use of Classifiers in Sequential Inference. In Proceedings of NIPS, pp. 995-1001.

Rahman A. and Ng V. (2010). Inducing Fine-Grained Semantic Classes via Hierarchical and Collective Classification. In Proceedings of the 23rd International Conference on Computational Linguistics (COLING) Beijing, China, August, pp. 931-939.

Riloff E. and Jones R. (1999). Learning dictionaries for information extraction by multi-level bootstrapping. In Proceedings of the 16th National Conference on Artificial Intelligence.

Riloff E. and Shepherd J. (1997). A corpus-based approach for building semantic lexicons. In Proceedings of the Second Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 117-124.

Robertson S.E., Walker S., Jones S., Hancock-Beaulieu M., Gatford M. (1995). Okapi at TREC-3. In Proceedings of the Third Text Retrieval Conference (TREC), pp.109-126.

Spärck Jones K., Walker S., & Robertson S. E. (2000). A probabilistic model of information retrieval: Development and comparative experiments. Information Processing and Management, 36(6), 779–808 (Part 1); 809–840 (Part 2).

Thelen M. and Riloff E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), USA, 214-221.

Titov I. and McDonald R. (2008). Modeling online reviews with multi-grain topic models. In Proceedings of the 17th international conference on World Wide Web. ACM, New York, NY, pp. 111-120.

Tsai R. T. and Chou C. (2011). Extracting Dish Names from Chinese Blog Reviews Using Suffix Arrays and a Multi-Modal CRF Model. In Proceedings of the ACM SIGIR First International Workshop on Entity-Oriented Search.

Vechtomova O. (2012). A semi-supervised approach to extracting multiword entity names from user reviews. In Proceedings of the ACM SIGIR 1st Joint International Workshop on Entity-oriented and Semantic Search (JIWES) 2012, Portland, Oregon, USA.

Vechtomova O. and Robertson S.E. (2012). A Domain-Independent Approach to Finding Related Entities. Information Processing and Management, 48(4), pp. 654-670.

Wang R.C. and Cohen W. (2009). Automatic Set Instance Extraction using the Web. In Proceedings of ACL-IJCNLP, Singapore.

Weeds J. and Weir D. (2003). A general framework for distributional similarity. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 81-88.

Weeds J. and Weir D. (2006). Co-occurrence retrieval: a flexible framework for lexical distributional similarity. Computational Linguistics, 31(4), 429-475.

Wiebe J., Wilson T. and Cardie C. (2005). Annotating expressions of opinions and emotions in language . Language Resources and Evaluation, 39, 165-210.

Yarowsky D. (1992) Word sense disambiguation using statistical models of Roget's categories trained on large corpora. In Proceedings of the Fourteenth International Conference on Computational Linguistics (COLING-92), pp. 454-460.

Yi J., Nasukawa T., Bunescu R., and Niblack W. (2003). Sentiment analyzer: Extracting sentiments about a given topic using natural language processing techniques. In Proceedings of the IEEE International Conference on Data Mining (ICDM).