Subjectivity Aware Conversational Search Services

Yacine Gaci
LIRIS - Université Lyon 1, France
yacine.gaci@univ-lyon1.fr

Jorge Ramírez
University of Trento, Italy
jorge.ramirezmedina@unitn.it

Boualem Benatallah
University of New South Wales,
Australia
& LIRIS - Université Lyon 1, France
b.benatallah@unsw.edu.au

Fabio Casati
University of Trento, Italy
& ServiceNow, USA
fabio.casati@gmail.com

Khalid Benabdeslem
LIRIS - Université Lyon 1, France
khalid.benabdeslem@univ-lyon1.fr

ABSTRACT

Online users are becoming increasingly dependent on Web services in choosing among products and services. This recent trend is motivated by the integration of conversational agents which took the human-machine interaction to unprecedented levels of ease, using natural language as a communication medium. Given the success of these systems, users are constantly switching to experiential search, providing utterances that are intrinsically subjective such as looking for a restaurant with a romantic ambiance, creative cooking or nice staff. Current Web services are unfortunately unable to decipher the subjective signals present in user utterances and only support objective attributes that are listed in service descriptions (e.g., restaurant address, cuisine, price range).

To make the most of dialog systems, they must be able to detect subjective attributes in user utterances and filter responses according to user subjective preferences. This paper presents a framework and techniques that augment conversational search services with capabilities to understand and reason about subjective user utterances. We propose novel subjective tag-based indexing of information services. We discuss automatic subjective tag extraction from both user utterances and online reviews using state of the art machine learning techniques such as BERT, adversarial training and data programming. Experiments show that the proposed techniques outperform existing information retrieval systems and the search mechanisms provided by well-known web search services such as Yelp.

1 INTRODUCTION

Digital services and online reviews are widely used in day-to-day decisions [14], such as providing recommendations or opinions regarding which restaurant to eat, which research paper to read or even who to vote for in elections. When we make decisions, recent studies show that we are prone to lean toward subjective data focusing on past experiences rather than relying solely on objective information (e.g., type of food served by a restaurant or an address of a restaurant) [31]. For example, when we set out to choose between two restaurants, we are attracted by the ones offering great experiences such as delicious food, brilliant atmosphere, friendly staff or romantic ambiance in addition to deciding based on factual information such as restaurant locations or specific types of cuisine [39]. Often, we find experiential

and subjective information in online reviews because they reflect user opinions and experiences [14]. Techniques from opinion mining and information retrieval (IR) [32] can be used to extract knowledge from reviews. However, such techniques usually lack the necessary precision to obtain meaningful and accurate subjective information due to their keyword-based search nature [31]. On the other hand, online reviews are expressed in natural language which is very nuanced and intricate, thus needing more effective extraction techniques. In the same line of argument, information retrieval systems are heavily manual for the users given that users need to try different combinations of keywords and query styles before having to compare between the results in a manual and labor-intensive way. Hence, decisions made through traditional information retrieval systems are generally sub-optimal [14].

Capturing and reasoning about subjective information has been explored at various levels [31]. Some techniques explored ratings (e.g., star ratings) to represent aggregated user opinions on entities or services [59]. Rating-based techniques do not consider reviews content but they rather provide aggregated numerical or symbolic values which are hard for users to express accurately. For example, a star rating of three out of five might give the impression that the restaurant is average in all aspects but in reality, it may serve delicious food but employs unhelpful waitstaff, which made the reviewer balance out her final rating. Another line of subjectivity-related research translates numerical attribute values to linguistic values (e.g., translation of prices to linguistic values, such as {"cheap", "fair", "costly", "expensive"}) using insights from fuzzy logic [26]. Nonetheless, such methods involve objective attributes, whose values are to be translated into subjective linguistic variations, leaving the space of the inherently subjective attributes untouched.

More recently, [31] proposed techniques extending database systems to account for both objective and subjective attributes and support subjective database queries. Nevertheless, in order to use [31], the database schema, and hence the subjective attributes, must be defined beforehand. Unfortunately, it is not always easy to identify which attributes to include in the schema and what their precise meaning is [14]. Besides, while such extensions [31] augment structured query languages with subjectivity support, they presuppose technical expertise comparable to that of professional database users, who can express (complex) SQL queries. Therefore, there is a need for more advanced techniques to empower all users to benefit from subjectivity-aware services in performing their day-to-day activities in a digitally enabled world.

At the same time, conversational Artificial Intelligence (AI) and its instantiation in the form of messaging or chat bots (also called task-oriented conversational bots) emerged as a new paradigm to naturally access services and perform tasks through natural language (text or voice) conversations with software services and humans. Thousands of bots have already been used in a variety of significant use cases, e.g. tourism, travel, office tasks, healthcare, e-commerce, education, and e-government services. On the downside, the current generation of conversational bots does not handle subjective information users ought to include in their utterances and often ignores them, leading to user dissatisfaction.

In this work, we propose *SACCS* (Subjectivity Aware Conversational Search Service), consisting of a Natural Language Understanding (NLU) framework and techniques, that combine the usefulness of including subjective information in the search utterances and the flexibility of utilizing natural conversations to interact with users. A key feature of *SACCS* is the ability to automatically and dynamically extract subjective information from user utterances and online reviews without explicitly defining them beforehand. Achieving such an objective faces several difficult issues, the most challenging of which is due to the expressiveness and complexity of natural language, i.e. the same subjective information can be expressed using various phrases. For instance, both *"The food is phenomenal"*, *"Very tasty plates of food"* or *"Really good food"* denote the deliciousness of food. To address this issue, we introduce the concept of *subjective tags*. Briefly stated, a subjective tag denotes subjective information in user utterances and online reviews. For example, the review sentence *"This restaurant serves really good food and the service is really quick"*, is tagged with {*delicious food*, *quick service*} subjective tags. The use of tags provides a powerful mechanism to reason about subjective information in user utterances and online reviews (e.g., organization, navigation, summarization, matching and understanding of subjective information). Building upon advances in opinion extraction, in our approach a subjective tag is represented as concatenation of an *aspect* term and an *opinion* term [32]. The aspect term denotes the feature being described and the opinion term characterizes this feature. For example, *delicious food* is a subjective tag wherein *food* is the aspect while *delicious* is the opinion. *SACCS* marks each review with a corresponding set of subjective tags.

In this paper we make the following contributions to overcome challenges related to extracting subjective tags from user utterances and online reviews as well as using them to support subjectivity-aware human-bot natural language conversations:

- We introduce a framework that augments task-oriented dialog systems with subjective filters. Search services are augmented with subjective tag based search and indexing [36]. Each subjective tag in the index is mapped to a list of reviews and entities (e.g. restaurants, books, hotels...).
- We provide a novel subjective tag extraction pipeline that is robust against variations of natural language. Tagging labels each word in a natural language sentence as being either an aspect, an opinion or neither. We train a subjective tag extraction model (called extractor) in an adversarial fashion, wherein the adversary [13, 38] adds *informed* perturbations to natural language sentences. This allows the tag extraction model to learn the possible variations in language and update its parameters accordingly.

- After the aspects and opinions have been extracted, there is a need to pair each aspect with its corresponding opinion in order to construct subjective tags. We propose two novel heuristics for pairing an aspect to an opinion. These heuristics aim to overcome the limitation of word-based distance approaches for pairing an aspect term to an opinion term [31, 56]. The first heuristic relies on the distance between aspects and opinions in the review parse trees [24, 25, 41]. For example, the opinion *professional* would be wrongfully paired with the aspect *decor* in the review *"The staff is friendly, helpful and professional. The decor is beautiful"* when relying on word distance alone. However, when using a parse tree to represent the above review, the two sentences *"The staff is friendly, helpful and professional"* and *"The decor is beautiful"* belong to two separate sub-trees. Consequently, the opinion *professional* will be closer to the aspect *staff* than the aspect *decor* because *professional* and *staff* belong to the same sub-tree. While more effective than traditional word distance techniques, this heuristic has the following limitations: (i) In long sentences, the different aspects and opinions may not be separated into their own sub-trees. In this case, this heuristic provides the same result as word distance methods. (ii) The generated parse tree will be incorrect when there are typos or punctuation errors in the review. The second heuristic is proposed to overcome these limitations. It relies on attention mechanism [3, 17, 54] to distribute the attention of an aspect term among the different opinion terms. These heuristics are combined using a data programming paradigm [2, 49] to : (i) pair opinion and aspect terms in natural language sentences in an unsupervised model, or (ii) automatically generate training data from online reviews to build a supervised pairing model.
- We evaluate the performance of the proposed techniques using crowd sourced data. Experiments show that *SACCS* provides better results than IR systems. Besides, the tagger improves upon state of the art by up to 4.93% in F1 scores while the supervised pairing method adds 3.03 points in accuracy.

This paper is organized as follows: We first discuss related work. We then introduce the architecture of *SACCS* in Section 3. The extraction of subjective tags is discussed in Sections 4 and 5. Section 4 describes the tagging while Section 5 details pairing. Finally, Section 6 discusses the evaluation of the proposed techniques.

2 RELATED WORK

Our work lies at the intersection of two areas: Subjectivity search, and aspect/opinion extraction.

Subjectivity Search. Despite the overwhelming importance of subjective information in the decision making process, relatively little effort focused on understanding and measuring the effect of subjectivity in user decisions [31]. This task has been traditionally delegated to standard information retrieval systems which provide a keyword-based search, and a synonym expansion at best [36, 52]. Systems that incorporate subjective filters in their data models attracted the attention of the research community only recently. Perhaps the closest work to ours is [31] which aimed to augment traditional database systems with subjective attributes. Their approach is different from ours in that their

subjective attributes are part of a database schema itself, which should be explicitly defined by a database designer beforehand. The query interfaces require the users to have precise knowledge about source schemas too. In our approach, subjective tags are dynamically extracted from user utterances as they interact with the system, thus increasing flexibility and productivity.

[27] built a tunable high-precision knowledge base with both factual and subjective attributes. To do so, they predefined a list of attributes (e.g. GOOD_VIEW, KID_FRIENDLY, HAS_HIGH_CHAIRS) and asked crowd workers to assess whether an entity (in their case, they used locations in Google Maps) has each attribute or not. They then modeled user consensus with Beta distributions. The major limitation of this approach is the increasing cost of crowd workers when adding new locations, new attributes or even changing the domain. Besides, crowd-sourced data suffers from data quality problems, mainly due but not limited to the inherent subjectivity in the task at hand. Also, the subjective attributes in [27] are set at design time and not learned from user interactions as we do.

Prior works also tackled the problem of subjectivity and opinions in various domains [29, 37, 59]. Most of them capture a narrow aspect of subjectivity by prompting the reviewers to rate the objects they write about. We often find these in e-commerce services in the form of star ratings which aggregate opinions of all sub-parts of the object and act as a proxy for the overall user satisfaction. This suffers from coarse granularity because the star rating skips the details we might be interested in and only gives one global assessment of the reviewer's true feeling.

Another body of research aims to translate objective facts into subjective phrases [20, 26, 50, 60]. The dominant example is the price which is mapped to a set of subjective phrases such as {"cheap", "fair", "costly", "expensive"} depending on comparisons between the price value and a set of thresholds. This approach only deals with translating objective attributes whose values are indisputable. It leaves the space of the inherently subjective attributes such as food deliciousness or room cleanliness untouched.

Aspect Opinion Extraction. The problem of extracting aspects from review texts is a long standing one in the Natural Language Processing (NLP) literature [32]. However, most previous work focused on identifying the aspects only and measuring their quantitative sentiment polarity (as being positive, negative or somewhere in between). This task is often referred to as *Aspect-Based Sentiment Analysis* (ABSA) [32].

Existing approaches include rule-based, feature-engineering-based and deep-learning-based approaches [56]. In a rule-based approach, to classify the aspect terms as positive or negative, a lexicon is used along with handcrafted sentiment values [18, 19]. Feature-based approaches [22, 30] train a classifier to extract the aspect terms with manually defined features. Both rule-based and feature-based solutions are labor-intensive and highly demanding in terms of effort and time. Deep-learning-based approaches [33, 55, 56], aside from having superior performance than the previous two methods, extend the extraction to opinion terms as well. While [55] used recursive neural networks, [56] employed an attention-based architecture. The motivation behind both approaches is the necessity to link aspects to opinions. [31] employed BERT sentence embeddings [7] with a standard classifier that classifies each word in the sentence into either Aspect, Opinion or Other. In the same spirit, we use BERT as an embedding layer along with a BiLSTM-CRF classification model. We also

Table 1: An example of an inverted index with degrees of truth for each subjective tag and restaurant pair

Tag	Restaurants
good food	Vue du Monde (0.89)
	Anchovy (0.76)
	Pizza Hut (0.82)
nice staff	Vue du Monde (0.92)
	Pizza Hut (0.63)
creative cooking	Anchovy (0.94)
	Pizza Hut (0.34)
	Kazuki's (0.85)
fast delivery	Anchovy (0.13)
	Pizza Hut (0.75)
	McDonald's (0.74)

leverage adversarial training to handle potential variations in the language. Experiments show that *SACCS* 's extractor yields better performance in various test benchmarks.

3 SUBJECTIVE TAG BASED INDEXING AND FILTERING IN CONVERSATIONAL SEARCH SERVICES

To describe the pipeline of *SACCS* , we begin with illustrating how subjective tags are used. We then move on to show how *SACCS* constructs these tags and how it uses them to answer complex and subjective user utterances. It should be noted that, while the proposed techniques are not domain specific, we choose the restaurants domain as a use case in this paper in order to illustrate the components of the proposed pipeline. We also assume that the underlying dialog system is already equipped with intent recognition [15, 23, 46] and slot filling techniques [4, 12]. Briefly stated, intent recognition allows the identification of user intents from user utterances. For instance, from the following user utterance: *"I want to eat Italian food near Lyon in a romantic ambiance"*, the dialog system identifies that the user is searching for a restaurant. Once an intent is identified, the system also extracts what is called slots (e.g., the type of cuisine (Italian), the location of the restaurant (Lyon)). The chatbot then delegates the search intent to a search API that retrieves a list of restaurants filtered by objective criteria. The goal of *SACCS* is to re-filter this list to only keep the restaurants which offer a romantic ambiance.

3.1 Subjective Tag Index

In order to use subjective tags, *SACCS* leverages an inverted index data structure [36]. Table 1 shows a snippet of what the index might look like [1]. Each subjective tag points to a set of entities (in this case restaurants) whose reviews include mentions of the subjective tag. For example, *good food* in Table 1 points to *Vue du Monde*, *Anchovy* and *Pizza Hut*, meaning that the reviews of these restaurants mention the deliciousness of the food cooked there. Also, every entity is accompanied by a degree of truth. Informally, a degree of truth associated to a tag measures the degree of certainty that *SACCS* exhibits when marking an entity with the tag. In the case of Table 1, *Vue du Monde* is more likely to have a nice staff than *Pizza Hut* (a degree of truth of 0.92 compared to 0.63). The degrees of truth are computed automatically by *SACCS* .

---

[1]The degrees of truth reported in the table are for illustration only and do not reflect the quality of these restaurants in the real world
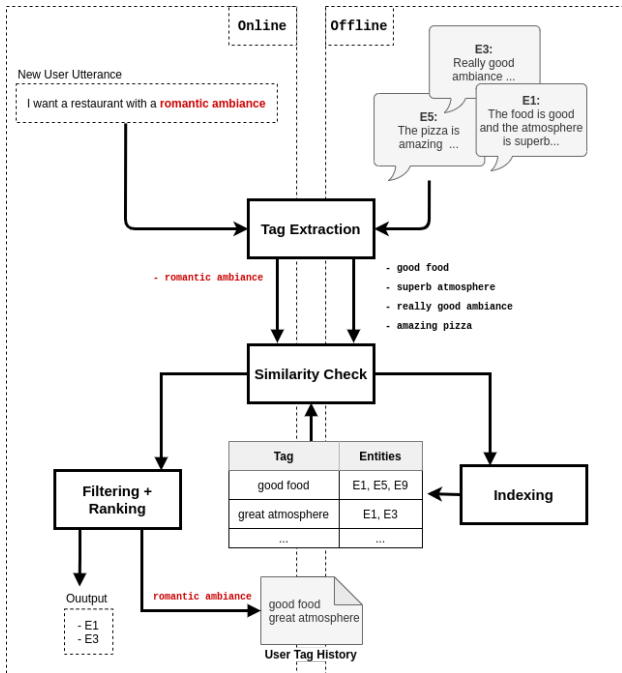
Figure 1: Architecture of *SACCS*

After collecting the set of subjective tags, *SACCS* needs to associate each tag with a set of entities, as depicted in Table 1. Association, or mapping, between a tag and an entity is based on similarity scores [53]: *SACCS* reads online reviews of the entity and extracts all subjective mentions from it. It then proceeds to compute similarities between the subjective tags in the index with those extracted from the reviews. If the similarity exceeds a predefined threshold, *SACCS* includes the corresponding entity to the index. Figure 1 illustrates this process.

The index in Figure 1 contains two subjective tags: *good food* and *great atmosphere*. Suppose we have three entities (*E1, E3* and *E5*) each having only one review. The extractor component extracts subjective tags from the reviews, in this case *good food, superb atmosphere, really good ambiance*. In the next step, the similarity checker computes similarity scores between the review tags and the index tags. Each time a similarity exceeds a specified threshold, the indexer adds the corresponding entity to the appropriate subjective tag in the index. Following the same example in Figure 1, *E1* and *E5* are both included as mappings to the subjective tag *good food* because their reviews both mention it (*good food* and *amazing pizza* for *E1* and *E5* respectively). However, the review of *E3* only mentions the ambiance; hence *SACCS* does not add it as a mapping to *good food*. We use conceptual similarity which, in addition to the individual meaning of words, also considers their nature or concept, for example pizza being a type of food [2]. Conceptual similarity has been shown to work better on short phrases such as subjective tags than cosine similarity. When building the index, *SACCS* automatically computes the degrees of truth of an entity *e* with respect to *tag*. The exact formula is shown in Equation 1.

---

[2]Conceptual similarity is outside the scope of this paper and may be subject to another submission

$$Deg\_truth(tag, e) = \frac{log(|R_e| + 1)}{|T_e^{tag}|} * \sum_{t \in T_e^{tag}} Sim(tag, t) \quad (1)$$

Where R_e is the set of entity *e*'s reviews and T_e^{tag} is the set of subjective tags automatically extracted from R_e and whose similarity score exceeds a predefined threshold θ_{index} when compared to the tag *tag*. $|R_e|$ and $|T_e^{tag}|$ are the number of elements in both R_e and T_e^{tag} respectively. Equation 1 finds all review tags which are similar to *tag* and computes the arithmetic mean of their similarity scores, weighted by the number of reviews. The motivation of multiplying the mean with the number of reviews for each entity is that the more reviews there are, the more statistically significant the degrees of truth become. That is why *SACCS* privileges the entities having more reviews.

Going back to the example in Figure 1, when the user submits a new utterance *"I want a restaurant with a romantic ambiance"*, *SACCS* extracts *romantic ambiance* from the utterance. Because this tag is unknown to *SACCS* , it adds it to the user tag history. Consequently, in the next indexing round, *SACCS* includes *romantic ambiance* to the index and computes its entity mappings along with their degrees of truth as has been explained above. This mechanism enables *SACCS* to adapt to new user needs.

3.2 Filtering

In this section, we provide details about how *SACCS* utilizes subjective tags to answer users subjective utterances.

Processing user utterances. Suppose the user submits a new utterance: *"I want an Italian restaurant in Melbourne that serves delicious food and has a nice staff"*. *SACCS* forwards this utterance to the underlying dialog system which finds the user intent (in this case *searchRestaurant*) and calls a corresponding search API (e.g. TripAdvisor, Yelp...). In this example, *SACCS* expects the API to return the set of restaurants that are in Melbourne and serve Italian food. We call this set S_{api}. As mentioned before, neither the dialog system nor the search API understand subjective information in the utterance such as *delicious food* and *nice staff*, thereby ignoring them completely. *SACCS* extracts these tags from the utterance and use them to filter and rank S_{api} before showing the final results to the user.

Probing the index. If the subjective tags extracted from the user utterance exist in the index, the corresponding entities with their degrees of truth are directly taken from the index. For instance, in the previous utterance, *nice staff* exists in the index depicted in Table 1, and thus the matching set ("Vue du Monde", 0.92), ("Pizza Hut", 0.63) is extracted as is. We call this set S_{t1}, where t1 = *"nice staff"*.

On the other hand, if the subjective tag is not found in the index, *SACCS* adds it to the user tag history as discussed in Section 3.1 and Figure 1 for later indexing. However, in order to provide a good answer to the user in real time, *SACCS* combines mappings of similar tags which are already in the index. To illustrate this, we go back to the previous example. *Delicious food* does not exist in the index of Table 1, but is similar to *good food* and *creative cooking*. In this case, *SACCS* calculates the union of the mappings corresponding to these two tags and multiply their degrees of truth by the similarity score of *delicious food* with each of the two subjective tags. Assume that:

$$s_1 = similarity(delicious\ food, good\ food) \quad (2)$$

$$s_2 = similarity(delicious\ food, creative\ cooking) \quad (3)$$

The set of entities that *SACCS* finds for *delicious food* is then $\mathcal{S}_{t2} = \{(\text{"Vue du Monde"}, s_1 \times 0.89), (\text{"Anchovy"}, s_1 \times 0.76 + s_2 \times 0.94), (\text{"Pizza Hut"}, s_1 \times 0.82 + s_2 \times 0.34), (\text{"Kazuki's"}, s_2 \times 0.85)\}$

After the construction of $\mathcal{S}_{api}$, $\mathcal{S}_{t1}$ and $\mathcal{S}_{t2}$, *SACCS* needs to aggregate the entities coming from the search API, plus the ones recovered from each subjective tag in the utterance. In other words, *SACCS* computes the **intersection** of these sets of entities according to Algorithm 1. It is worth noting that the function *search_api* takes the user utterance as input parameter and relies on the underlying dialog system and the search API to provide results filtered by objective attributes alone. On the other hand, the function *extract_tags* takes the user utterance as input parameter and returns the list of subjective tags using the extraction pipeline which we describe in Sections 4 and 5.

Algorithm 1 Filtering & Ranking

1: Let **u** be the user utterance
2: Let **index** be the inverted index
3: Let θ_{filter} be the similarity threshold
4: $\mathcal{S}_{api} \leftarrow search\_api(u)$
5: $tags \leftarrow extract\_tags(u)$
6: **for** t in tags **do**
7:     **if** $t \in index.keys$ **then**
8:         $\mathcal{S}_t \leftarrow index[t]$
9:     **else**
10:         $\mathcal{S}_t \leftarrow \bigcup_{tag \in index.keys} \{index[tag]\}$ such that $similarity(t, tag) > \theta_{filter}$
11: $\mathcal{R} \leftarrow \bigcap_{t \in tags} \{\mathcal{S}_{api}, \mathcal{S}_t\}$
12: **Return** $sort(aggregate\_scores(\mathcal{R}))$

3.3 Ranking

SACCS ranks the filtered set of entities according to their degrees of truth across all subjective tags. We identify two situations for ranking.

One subjective tag. If the user expresses a single subjective filter in her utterance, the ranking is straightforward. *SACCS* sorts the entities according to their degrees of truth in descending order, so that the top results are the ones whose reviews strongly mention the subjective tag.

Many subjective tags. In this case, *SACCS* has a separate set of entities with their degrees of truth for each subjective tag. However, an entity can belong to many of such sets. Thus, before ranking becomes feasible, *SACCS* must aggregate the degrees of truth for each entity across all subjective tags. Aggregation is done via computing the arithmetic mean over all tags. We also experimented with other aggregation methods such as the product or min operators, but the arithmetic mean works better in practice. *SACCS* then sorts the entities in descending order. Algorithm 1 combines the filtering and ranking stages. In line 12, the function *aggregate_scores* computes the arithmetic mean of degrees of truth across the tags.
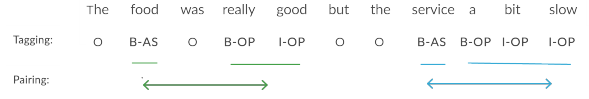

Figure 2: Token Tagging and Pairing

As mentioned before, a subjective tag is the concatenation of two terms: the **aspect** term and the **opinion** term. Following previous effort [31], we formulate the task of extracting subjective tags from a given input sentence as a two-stage process: tagging and pairing, as illustrated in Figure 2. Each word in the sentence is first tagged as being an aspect (AS), an opinion (OP) or neither (O). Then, every aspect term gets paired with its corresponding opinion term to build the set of subjective tags from the input sentence. In the following, we describe the techniques we propose for tagging and pairing tasks. We describe tagging in Section 4 and pairing in Section 5.

4 TAGGING

We denote by r_i a review sentence which consists of a sequence of tokens $r_i = \{w_{i1}, w_{i2}, ..., w_{in}\}$. We use the IOB encoding scheme [47] with the following classes: B-AS (Beginning of Aspect), I-AS (Inside of Aspect), B-OP (Beginning of Opinion), I-OP (Inside of Opinion) and O (Outside). The set of tags is thus $L = \{$B-AS, I-AS, B-OP, I-OP, O$\}$. The objective of tagging is to classify each token w_{ij} in the sentence r_i, into a class $c_{ij} \in L$. The components of *SACCS*'s tagging model are detailed below.

4.1 Baseline for the Tagging Pipeline

Figure 3 depicts the base architecture for tagging words into aspects and opinions. We use BERT [7], the recently-developed language model, as the embedding layer thanks to its proven superior quality when compared to other embedding models [7]. As illustrated in Figure 3, BERT embeddings serve as input to the Bidirectional LSTM (BiLSTM) layer [16], which encodes the past context (all words prior to any given word in the sentence) and the future context (all words following a given word) of each word. Following [8, 35], we encode the text sequence from both left to right (forward) and right to left (backward). We then concatenate the resulting representations to form the final output of the BiLSTM.

Finally, the BiLSTM output flows to the Conditional Random Field (CRF) layer [28], which is paramount to encode dependencies between the different labels of L. For example, I-OP cannot follow I-AS in the label sequence. More generally, I-AS (or I-OP) must either follow B-AS or I-AS (B-OP or I-OP). Given an input sequence $z = \{z_1, z_2, ..., z_n\}$, CRFs effectively utilize correlations between labels to predict the best label sequence $y = \{y_1, y_2, ..., y_n\}$. Formally, the conditional probability function of CRFs is given by:

$$P(y|z, W, b) = \frac{\prod\limits_{i=1}^{n} \psi_i(y_{i-1}, y_i, z)}{\sum\limits_{y' \in Y(z)} \prod\limits_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, z)} \quad (4)$$

where $Y(z)$ denotes the set of possible labels for the sequence z and $\psi_i(y_{i-1}, y_i, z) = exp(W_{y', y}^T z_i + b_{y', y})$ are potential functions to be learned with $W_{y', y}$ and $b_{y', y}$ being the weight and bias vectors respectively. Decoding (i.e. solving the tagging task using
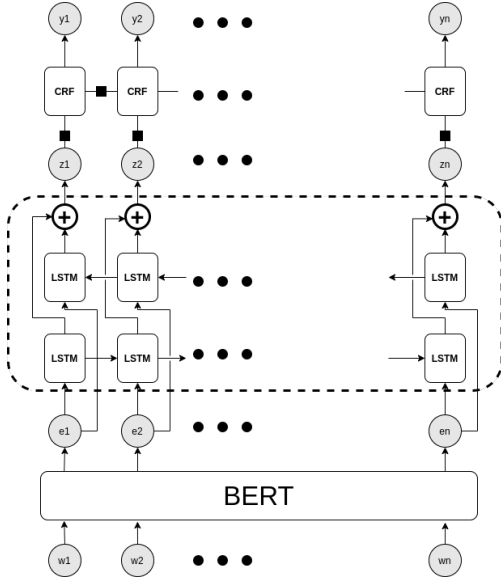
Figure 3: Sequence tagging model based on BERT + BiL-STM + CRF

a CRF layer) consists in finding the best sequence of labels y that maximizes the log-likelihood given the input sequence z:

$$y^* = \underset{y' \in Y(z)}{\text{argmax}} \, P(y'|z, W, b) \tag{5}$$

In this work, we use linear-chain CRFs, where only interactions between two successive labels are taken into consideration. We also adopt the Viterbi algorithm [10] along with beam search for efficient decoding of the label sequence.

4.2 Extending the Baseline with Domain Adaptation

In *"La carte of this restaurant is a killer"*, *SACCS* should be able to tag *la carte* as an aspect and *a killer* as an opinion. However, opinions are mostly adjectives whereas *a killer* is a noun, thereby *SACCS* might fail to recognize it as an opinion, or even mark it as an aspect. Moreover, *la carte* is a rare word in the english vocabulary, thus the tagger might not understand the word altogether. This limitation is largely due to the fact that BERT has been pre-trained on general Wikipedia articles [7]. As a consequence, it does not know that *a killer* is a widely used idiom in the restaurant jargon to characterize something as overly good. It also ignores that *la carte* in this case means *the menu*, which is an important aspect to be extracted. Hence, standard BERT embeddings are blind to the domain and may hinder the tagging performance of *SACCS* .

To make the embeddings more domain-aware, we follow the guidelines of [58] who post-trained BERT on domain-specific review corpora in order to make it understand opinion text rather than generic Wikipedia articles. We use reviews about restaurants as a post-training dataset in our case. [58] also added another fine-tuning iteration to make BERT aware of the task (e.g. aspect/opinion extraction), but on out-of-domain data. We find that using domain knowledge alone works better in our case than when leveraging both domain- and task-awareness. Experiments show that domain adaptation adds up to 2.93 F1 score points over the baseline.

4.3 Adversarial Learning for Dealing with Language Expressiveness

Natural language is very nuanced, and introducing subtle changes to the input sentences can change the meaning dramatically. For example, adding *not* before the verb or changing *always* with *never* reverse the meaning of the sentences completely. Unfortunately, such changes happen frequently when using the trained model with new sentences, unseen during training. This is particularly alarming when the changes are subtle, or insignificant when assessed by a human evaluator, for example changing a word with its synonym. However, word embeddings do not always align with human perception. For instance, two synonymous words might be far apart in the embedding space [9, 21, 40]. Even if they are close to each other, the tiny distance between the embeddings can be enough to mislead the trained model [38]. Adversarial examples have long been used to make trained models robust against small input differences and perturbations (noise). It has been shown to provide additional regularization capabilities beyond that brought by the use of dropout alone [13].

We leverage adversarial learning to enhance the robustness of *SACCS* 's tagger against input noise. We generate adversarial examples that are close to the original inputs and that should share the same label sequence (i.e. aspect/opinion tags), yet are specifically designed to fool the model into tagging them otherwise. The creation of these adversarial inputs is enabled by the introduction of small *worst case* perturbations bounded by a chosen perturbation set, to decrease the model's ability to predict correctly. The tagger is then trained on a mixture of clean and adversarial examples to enhance its stability and robustness against potential input perturbation. The objective function is thus the following:

$$\underset{\theta}{\text{Min}}[\alpha.l(h_\theta(x), y) + (1 - \alpha). \underset{\delta \in \Delta(x)}{\text{Max}} \, l(h_\theta(x + \delta), y)] \tag{6}$$

where h_θ is the tagging model with θ being the corresponding parameters. l is the loss function and $\Delta(x)$ is the set of perturbations allowed for the input sequence x. In this work, we use the l_∞ ball: $\Delta(x) = \{\delta : ||\delta||_\infty < \epsilon\}$ where ϵ is a hyperparameter to be tuned. Equation 6 assumes the perturbations to be applied directly on the embeddings as has been done in [38]. Solving such an objective function exactly is intractable in complex networks. Consequently, by leveraging Danskin's theorem [6], we can first solve the inner maximization independently to find δ^* that maximizes the adversarial loss, and then adding δ^* to the input to solve the outer minimization objective.

$$\delta^* = \underset{||\delta||_\infty < \epsilon}{\text{argmax}} \, l(h_\theta(x + \delta), y) \tag{7}$$

$$\underset{\theta}{\text{Min}}[\alpha.l(h_\theta(x), y) + (1 - \alpha).l(h_\theta(x + \delta^*), y)] \tag{8}$$

Finding an exact solution for δ^* is also an intractable problem for complex models. We approximate δ^* by assuming a *linear* tendency for the adversarial loss inside the norm-ball. We thus use the Fast Gradient Sign Method (FGSM) suggested by [13] to find a decent solution in an efficient way. The computation of δ^* is given by:

$$\delta^* = \epsilon.sign(g) \tag{9}$$

where $g = \nabla_\delta l(h_\theta(x + \delta), y)$. In Equation 8, the first loss is the clean loss, while the second loss represents its adversarial counterpart. The parameter α denotes how much weight we
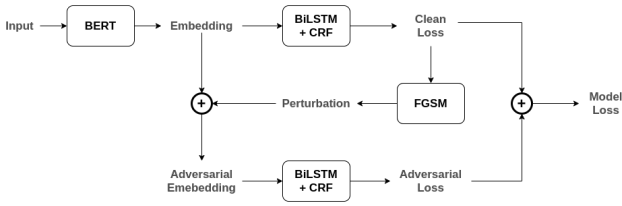
Figure 4: Architecture for Adversarial learning using BERT

Figure 5: BERT attention head for pairing aspect and opinion terms

give to the adversarial example with respect to the original one. Figure 4 illustrates the entire adversarial learning component.

5 PAIRING

In Figure 2, *food* is paired with *really good*, and *service* with *a bit slow* [3] in order to create the corresponding subjective tags. Most previous work [55, 56] employ simple heuristics such as word distance to pair aspects and opinions. However, such techniques fall short of the expected accuracy especially on complex and tricky input sentences. For example, the opinion *professional* would be wrongfully paired with the aspect *decor* in the review *"The staff is friendly, helpful and professional. The decor is beautiful"* when relying on word distance alone, because *professional* is closer to *decor* than to *staff*.

In this section, we first describe the two novel heuristics that we propose to pair aspects with opinions (Section 5.1). Although these heuristics can be directly used as an unsupervised pairing model, in Section 5.2, we discuss how they are used in a supervised model.

5.1 Pairing Heuristics

We design two types of unsupervised heuristics for pairing. The first category is based on constituency parse trees [24, 25, 41] while the second utilizes the attention mechanism [17].

Heuristic based on parse trees. The first method is a rule-based method. The intuition behind it is that associated aspects and opinions should be close to each other in the parse tree of the input sentence. We start by building the parse tree and then apply a greedy strategy that maps every aspect term to the "closest" opinion term in the parse tree. Given that a single aspect can be mapped to multiple opinions [4], we use this heuristic twice: from aspects to opinions and then from opinions to aspects. For example, in *"The staff is friendly and professional"*, *friendly* is closer to *staff* than *professional* is in the parse tree. Hence, the first version outputs the pair (*staff*, *friendly*). On the other hand, the second run starts from opinions and looks for the closest aspect. It would thus give the pairs (*staff*, *friendly*) and (*staff*, *professional*).

Heuristic based on BERT attention heads. The idea behind using BERT attention heads for pairing is motivated by the need to assign relevance scores to aspects and opinions. Ideally, we want each aspect term to focus more on its corresponding opinion (high relevance score) and ignores the rest (low relevance scores). Attention can be leveraged to approximate relevance. First introduced to enhance neural machine translation
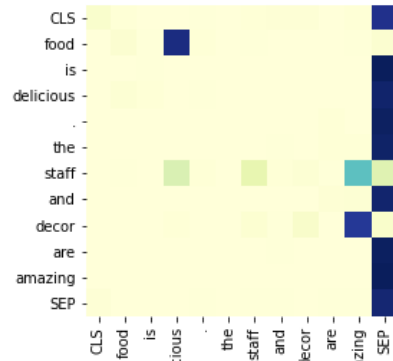
[3] and later adapted to nearly every other NLP task, attention is a mechanism to assign importance values to every token in the sequence given a query term. We say that the query term attends to the tokens which have the highest attention scores. In our case, the query term is the aspect term, and the sequence is the input sentence. Using this method, the goal is to distribute the attention of every aspect term so that it attends to the rightful opinion (that of the highest attention).

Fortunately, BERT is an attention-based model, and we have it already trained on aspect/opinion extraction as explained in Section 4. We hypothesize that, while learning the downstream task of tagging aspects and opinions, BERT leverages its attention heads in a way that makes aspects attend to the rightful opinions, and vice versa. Figure 5 confirms our hypothesis. It illustrates one attention head of BERT. Each row in the figure is the attention distribution of the corresponding word over the entire input sequence; the darker the color, the higher the attention. In the figure, *food* is darkest at *delicious*, meaning that *food*'s attention to *delicious* is very high. In the same spirit, both *staff* and *decor* attend to *amazing*. Thus, BERT attention heads act as simple no-training-required classifiers that, given an aspect, output the most attended to opinion. We find that BERT heads capture various linguistic properties, some of which correspond remarkably well to the notion of pairing aspects with opinions. The best head we found for pairing has an accuracy of 82.62% on the pairing test set (Section 6.4), which is excellent given the quasi-none effort this method needs.

5.2 Supervised Learning-based Approach for Pairing

We also provide a supervised learning alternative to the problem of pairing. We formulate our pairing objective as a classification problem. Given an input sentence s_i (e.g., *"The food is delicious and the staff are friendly"*) and a short phrase p_i (e.g., *"delicious food"*)[5], the classifier classifies p_i as being a correct extraction from s_i or not. To use this classifier with *SACCS*, we first use the tagger to extract aspects and opinions from s_i. We then construct all possible pairs from the sets of aspects and opinions regardless of their soundness. For example, suppose we have *food* and *staff* as aspects, and *delicious* and *friendly* as opinions. The list of all possible pairings is: P_{all} = [*"delicious food"*, *"delicious staff"*,

---

[3]A multi-word aspect (or opinion) is regarded as a single aspect (opinion) term
[4]The reverse also applies: An opinion term can be paired with multiple aspects as well

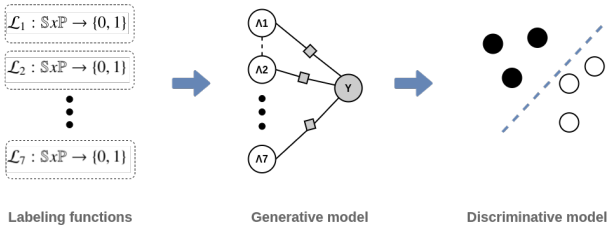[5]In the context of this work, these short phrases are subjective tags

Figure 6: Data Programming pipeline for pairing

"friendly food", "friendly staff"]. We feed s_i with each pair from P_{all} into the classifier, and consider it as a correct extraction if the classifier returns a positive label.

We use data programming [2, 49] in order to create the dataset necessary to train such a classifier. The entire pipeline is illustrated in Figure 6. First, a set of labeling functions [2, 48, 49] use the heuristics described in Section 5.1 in order to independently assign a label to every (s_i, p_i) pair. These labeling functions are considered weak supervision sources.

The second step in the pipeline aggregates the labels from the labeling functions to construct a single overall label for every (s_i, p_i) pair, based on agreements and disagreements between labeling functions. This is generally achieved with generative models which, by aggregating enough datapoints, end up creating a decent labeled training dataset. Finally, we use this dataset to train a discriminative model. In our case, it's the classifier discussed above. It is important to note that, in our case, we have a working solution for pairing aspects with opinions at each step of the pipeline. However, experiments show that committing to the entirety of the pipeline and using the discriminative model (and thus the supervised model) drives a considerable boost in pairing accuracy when compared to the unsupervised methods.

In the following, we describe the labeling functions, the generative and discriminative models we use in the supervised learning-based pairing pipeline.

Labeling functions for the pairing pipeline. A labeling function in *SACCS* 's pairing module has the same interface as the classifier, i.e. expects a sentence s_i and a phrase p_i as input, and outputs a binary label telling whether p_i is a legit extraction from s_i. All labeling functions are based on the heuristics presented in Section 5.1. To transform each heuristic H_j into a labeling function L_j, we follow the procedure below:

(1) Extract all aspects and opinions from s_i using *SACCS* 's tagger.
(2) Use H_j to find the pairs $P_{H_j^i}$ as detailed in Section 5.1.
(3) If the short phrase p_i belongs to the set of constructed pairs $P_{H_j^i}$, output 1. Otherwise, return 0.

We use seven different labeling functions: two are based on the parse tree method (the first from aspects to opinions, the second the other way around) while the remaining five rely on BERT attention scores employing different heads. The choice of attention heads has been made after a qualitative analysis.

Generative model for the pairing pipeline. We use the generative model proposed by Snorkel [48] in our pipeline. Snorkel is a data programming framework that integrates the noisy signals of multiple labeling functions to estimate the true label class[48]. Snorkel offers two mechanisms for aggregation. The simplest is a majority vote model where each labeling function is

regarded as an independent voter. The chosen label for each datapoint is the most agreed upon by labeling functions. The other method incorporates statistical properties of labeling functions such as accuracies and correlations. Snorkel then trains a probabilistic graphical model to *generate* the true labels without access to ground truth data. Training is based on agreements and disagreements between the different labeling functions as dictated by data programming. Although the authors of Snorkel state that the probabilistic generative model works better in practice than the majority vote, we found the latter to be more accurate.

We can directly use the generative model to extract subjective tags from review sentences. However, a better use of data programming lies in the automatic creation of labelled training data to train a subsequent discriminative model. The advantages of doing so are twofold: First, the discriminative model generalizes beyond the scope of examples fed to the labeling functions. Second, the discriminative model is faster to execute because the generative model loops through all labeling functions and aggregates their outputs, whereas the discriminative model only uses one forward pass in case of neural networks.

Discriminative model for the pairing pipeline. We train a simple two-layer neural network with a sigmoid activation function. We encode s_i and p_i using BERT embeddings. We train the classifier with the training data that has been automatically created with the procedure explained in the previous sections. Our experiments confirm that the discriminative model outperforms the generative one, as has been found in [48].

6 EXPERIMENTS

We first begin by showing our experimental settings before describing the experiments that we conducted. The first set of experiments evaluates the overall performance of *SACCS* and compares it to two baselines (Section 6.2). We then move to assess the quality of *SACCS* components. We evaluate the sequence tagger in Section 6.3 and the pairing mechanism in Section 6.4.

6.1 Experimental Settings

Datasets. We apply *SACCS* to the domain of restaurants whose online reviews we get from the publicly available Yelp Dataset [1]. Since it covers a wide array of businesses, we filter it to only keep reviews about Italian restaurants in Montreal, resulting in 280 entities (restaurants) with 7061 reviews. To train the aspect/opinion tagger, we use the training dataset created by [31] that contains 800 sentences, wherein each token is accompanied by its label. For pairing, we use the same training dataset as in [31] but without the labels since we augment the data and infer the labels with Snorkel [48].

Processing. We implemented *SACCS* in Python using standard packages such as PyTorch [42] for neural networks, HuggingFace transformers library [57] for BERT, NLTK [34] for textual preprocessing and Scikit-Learn [43] for evaluation metrics. In order to incorporate domain knowledge into BERT, we directly use the models for restaurants published on Huggingface community hub by [58]. For adversarial training, we fix the value of α to 0.5 (Equation 8) while we vary ϵ between {0.1, 0.2, 0.5, 1.0, 2.0}.

6.2 Comparing *SACCS* with Baselines

In this experiment, we evaluate the overall performance of *SACCS* , and then compare it to two strong baselines. The evaluation works as follows: we first prepare a set of subjective tags as a test set. Each system that we want to evaluate takes the tags as input and returns an ordered list of results, sorted by their degree of relevance with respect to the subjective tags. The result of each system is then compared to the ideal ordering of entities. The system whose ordering is "closest" to the ideal one is deemed the best. We apply this experiment to the domain of restaurants.

Preparing subjective tags. Since there is no benchmark for subjective tags, we had to create our own. [39] identified the most important features restaurant seekers consider when choosing a restaurant. These features include *"delicious food"*, *"creative cooking"*, *"varied menu"*, *"romantic ambiance"*... We chose 18 of them to serve as our subjective tags for testing purposes. We then construct combinations of these tags by uniform random sampling. Each combination will form a potential subjective user utterance. For example, if random sampling puts together the tags *"clean plates"* and *"quick service"*, it works as if a user gave the following utterance to the system: *"I am looking for a restaurant that delivers a quick service with clean plates."*. The number of tags per combination depends on the level of difficulty of the query (utterance). In this experiment, we set 3 levels of difficulty: Short with either 1 or 2 tags; Medium with 3 or 4; Long with 5 or 6 tags. Each set (level of difficulty) contains 100 queries (combinations).

Evaluation metrics. To measure how well the entities returned by *SACCS* and the baselines satisfy the queries in the test set, we use the well-known Normalized Discounted Cumulative Gain (NDCG) [5] which is a measure of ranking quality. Formally, this metric computes the quality of a ranked list and divides it by that of the ideal ordering, thus giving a score between 0 and 1, the higher the better. For illustration purposes, assume that subjective query Q has n subjective tags: $Q = \{q_1, q_2, ..., q_n\}$ and that we input Q to *SACCS* . The latter returns a list of top-k entities $E = \{e_1, e_2, ..., e_k\}$. We define $sat(q_i, e_j) \in [0, 1]$ as the degree with which entity e_j satisfies the subjective tag q_i. The NDCG score is computed as follows:

$$DCG(Q, E) = \sum_{j=1}^{k} (2^{\frac{1}{m}\sum_{i=1}^{m} sat(q_i, e_j)} - 1)/log_2(j + 1) \qquad (10)$$

$$NDCG(Q, E) = DCG(Q, E)/iDCG(Q) \qquad (11)$$

Intuitively, a highly relevant entity (that with $sat(q_i, e_j)$ scores close to 1) should be at the top in order for the DCG to be high. iDCG in Equation 11 corresponds to the DCG score of the ideal ordering. It is fairly easy to get the iDCG as it is only a matter of sorting the entities with respect to the sum of their $sat(q_i, e_j)$ scores and then computing the DCG. Finally, we take the arithmetic mean over all queries to compute the quality of the entire test set.

Ground truth. We obtain the ground truth $sat(q_i, e_j)$ of subjective tag q_i and entity e_j via crowdsourcing. We give each worker a tag q_i and one review r_j^k from the set of online reviews corresponding to entity e_j. The crowdsourcing task is to inspect the review r_j^k and tell whether it mentions the tag q_i or not. The worker must assign each pair of review/tag a relevance score

Table 2: Comparing *SACCS* to baselines

System	Short	Medium	Long
IR	0.829	0.896	0.916
SIM - 1 att	0.828	0.886	0.907
SIM - 2 atts	0.837	0.891	0.909
SACCS - 6 tags	0.815	0.874	0.896
SACCS - 12 tags	0.825	0.882	0.902
SACCS - 18 tags	**0.854**	**0.911**	**0.928**

among the following: 0 for no relevance, $\frac{1}{3}$ for weak relevance, $\frac{2}{3}$ for strong relevance and 1 for perfect relevance. As an example, given the review sentence *"The food is very delicious but the service is terrible"*, the tag *great food* should be marked as perfectly relevant, *nice decor* not relevant while *slow service* as weakly relevant because the slowness of the service is somewhat related to it being terrible. For each review/tag pair, we ask three different workers to provide labels, from which we take the majority vote, resulting in $sat(q_i, r_j^k)$ relevance scores. To obtain $sat(q_i, e_j)$, we take the mean of $sat(q_i, r_j^k)$ across the reviews of the same entity e_j. The crowdsourcing experiment has been conducted on Yandex Toloka platform[6].

Baselines. We compare *SACCS* to two baselines: an Information Retrieval (IR) system and a custom simulation (SIM). The IR baseline uses Okapi BM25 [5] retrieval model. We follow the work of [11] and add the capability to expand the terms of the query into synonymous and related terms, as well as select the best query combination method they found to make the IR system more competitive.

SIM represents what a determined and tireless user can get from Yelp or other similar online services. Because these services provide a set of queryable attributes (such as NoiseLevel, Ambiance or GoodForGroups), the user might filter the search results with the attributes she thinks closely resemble her subjective preferences. For example, if she is interested in quiet restaurants, she can set the attribute NoiseLevel to *calm* and the attribute GoodForGroups to *False* in Yelp's interface. She can also rank the results by star rating. SIM is a simulation of such behavior. We assume that the user can choose one or two attributes from Yelp's interface at a time. SIM computes all possible combinations of attribute values and selects the one that maximizes the NDCG score, thus finding the best top-k results that satisfy the subjective queries. It's needless to say that SIM constitutes a very strong baseline to compare *SACCS* against.

Comparison and analysis. Table 2 reports the NDCG scores of the three systems on the test set. Each column corresponds to the level of difficulty, that is Short, Medium or Long. The first row shows the quality of the IR system. The following two lines are variations of SIM using only one attribute, or a combination of two separate attributes. The last 3 rows describe the performance of *SACCS* , each time with a different number of subjective tags present in the index. This is to simulate the adaptive capability of *SACCS* as interactions with users unfold.

In all difficulty levels, *SACCS* outperforms the information retrieval system with a margin between 1.2% and 2.5%. This is not surprising because the IR system is based on keywords and looks for exact match whereas *SACCS* models subjective

---

[6]https://toloka.yandex.com

Table 3: Dataset Descriptions with number of sentences for train and test

Dataset	Description	Train	Test	Total
S1	SemEval-14 Restaurants	3041	800	3841
S2	SemEval-14 Electronics	3045	800	3845
S3	SemEval-15 Restaurants	1315	685	2000
S4	Booking.com Hotels	800	112	912

Table 4: Evaluation of aspect/opinion tagger

Models	S1	S2	S3	S4
OpineDB	81.82	75.44	72.30	67.41
OpineDB + DK	83.06	75.42	73.86	69.64
Adversarial ($\epsilon = 0.1$)	81.23	76.56	**74.63**	70.16
Adversarial ($\epsilon = 0.2$)	83.46	**76.97**	73.64	**72.34**
Adversarial ($\epsilon = 0.5$)	**84.43**	75.36	72.28	70.32
Adversarial ($\epsilon = 1.0$)	82.80	67.50	73.47	70.38
Adversarial ($\epsilon = 2.0$)	82.93	71.39	73.27	68.42

attributes with subjective tags. Table 2 shows that *SACCS* is superior to keyword-based systems even when the latter are bulked with query expansion and adequate predicate aggregation techniques. On the other front, SIM simulates the behavior of a determined user that runs through all possible combinations of queryable attributes that online services such as Yelp offer. To make the evaluation challenging, we take the combination that maximizes the NDCG score, thus reflecting the best result a user can have when interacting with Yelp's interface. As shown on the table, considering two attributes yields better results than one attribute, but with diminishing returns. That is why we don't bother searching the space of more than two attributes, which adds a non-negligible amount of computation. *SACCS* outruns SIM with 2 attributes by a margin between 1.7% and 2.0%.

Even with a small number of tags in the index, the performance of *SACCS* is comparable to that of IR or SIM. This is especially the case at the initialization of the index, where it finds itself nearly empty (in Table 2, the index contains 6 tags only). However, as *SACCS* interacts with users, it extracts new subjective tags from user utterances and adds them to the index in a dynamic and adaptive way. This experiment demonstrates that adding more tags to the index improves the overall accuracy (improvement between 3.2% and 3.9%), and confirms that *SACCS* adapts to new user needs.

We also observe that, for all three systems, accuracy increases with a higher number of subjective criteria. We hypothesize that with more subjective tags, the list of restaurants which verify all the subjective filters shrinks, leading to a lower margin for error in all systems; thus a higher NDCG score. Nonetheless, *SACCS* is still the best no matter the number of subjective tags to be considered. We also observe that the largest improvement happens with short queries (1 or 2 subjective tags therein). This result reinforces the integration of *SACCS* to task-oriented dialog systems where utterances are short and usually span a small number of subjective filters.

6.3 Sequence Tagging Evaluation

We show that the aspect and opinion tagger of *SACCS* is of better quality than that of state of the art, especially when the training dataset is small. We evaluate the sequence tagging model with 4 different datasets summarised in Table 3. The first three datasets are from SemEval competitions: SemEval 2014 Task 4 (Restaurants and Electronics)[45] and SemEval 2015 Task 12 (Restaurants)[44]. Each dataset contains a set of sentences where each token is labeled as being an aspect, an opinion or neither, following IOB coding scheme [47]. The original SemEval datasets contain labels for aspects only. However, we use the versions of [31, 55, 56] who added labels for opinions to the original sentences. The last dataset has been created and labeled by [31]. The goal of this experiment is to compare *SACCS* 's tagger with the strongest previous works in the literature, using datasets of different sizes and domains as well.

Other extraction tasks such as Named Entity Recognition (NER) [51] employ F1 to measure the quality of tagging. In the same spirit, Table 4 reports F1 scores of the extraction quality. For an aspect (or opinion) to be counted as correctly extracted, it needs to match the exact terms present in the ground truth. We compare our tagger against two strong baselines: OpineDB's tagger [31] which is a BERT-based solution that outperformed previous works in the literature [55, 56] and currently enjoys state of the art performance. We also enhance OpineDB's tagger with the domain-specific fine-tuning strategy suggested by [58] to make it even stronger (**OpineDB + DK** in Table 4). We evaluate our adversarial tagging model with different sizes of perturbations (ϵ values) as shown in Table4, but we fix $\alpha = 0.5$ (Equation 8) across all runs. The models are trained for 15 epochs.

The adversarial tagging model beats state of the art performance in all four datasets with an improvement ranging from 1.53% to 4.93%. As shown in the table, fine-tuning BERT with domain knowledge (DK) improves the performance by up to 2.23%, confirming the findings of [58] on aspect-based sentiment analysis. However, the boost of domain fine-tuning is not enough to outperform the adversarial training component, motivating the integration of adversarial examples in deep learning models. We also note that the adversarial component works better for smaller datasets (S4). We believe this is due to the regularization capabilities adversarial training provides as a counter-measure against overfitting beyond what is already ensured with dropout. We also notice that the tagging model performs best with lower perturbation sizes ($\epsilon \in \{0.1, 0.2, 0.5\}$), in which case the adversarial examples remain "closer" to the original ones, in contrast to large perturbation sizes ($\epsilon \in \{1.0, 2.0\}$) that can lead the model to exhibit poor accuracy. The issue of large ϵ values is especially noticeable with the Electronics dataset (S2) where $\epsilon = 1.0$ makes the adversarial model worse than OpineDB's baseline. We hypothesize that, since the Electronics dataset contains many technical terms such as brand names and numerical references, adding slight perturbations can change the meaning of terms completely while keeping the same labels, leading to the model's poor performance.

These results are very promising in the context of task-oriented dialog systems. Since chatbots should cover a wide array of domains, they need to be trained and fine-tuned for every single one of them. This task implies the creation of various datasets that are large enough to ensure a decent learning. Fortunately, Table 4 shows that our tagging model is efficient even with small training data (S4), thus eliminating the need to build large and costly datasets.

Table 5: Evaluation of the pairing models

Models	Accuracy	Precision	Recall	F1
OpineDB	83.87	/	/	/
lf_bert_7:10	82.62	95.02	78.36	85.89
lf_bert_3:10	74.56	91.54	68.66	78.46
lf_bert_3:8	68.26	91.76	58.21	71.23
lf_bert_4:6	75.82	93.00	69.40	79.49
lf_bert_8:9	77.33	94.95	70.15	80.69
lf_tree_op	74.06	92.31	67.16	77.75
lf_tree_as	76.07	91.00	71.64	80.17
Majority Vote	84.10	97.20	78.70	87.00
Probablistic Model	82.40	**98.10**	75.40	85.20
Discriminative	**86.90**	92.52	**87.69**	**90.04**

6.4 Pairing Evaluation

In this section, we evaluate the accuracy of the pairing model. We use the test benchmark created by [31] and employed to conduct their own experiments. Each test example consists of a review sentence (e.g., *"The food is delicious and the staff is helpful"*), a tag (*"delicious staff"*) and the label is whether the tag is a correct extraction from the review sentence. The test set contains 397 sentences with a fairly equal amount of positive and negative examples. We compare the accuracy of *SACCS* 's pairing model with that of [31] in Table 5. To highlight the effectiveness of data programming in the context of pairing and motivate the use of both generative and discriminative models, we also assess the quality of every step in the data programming pipeline presented in Section 5.2. Thus, Table 5 reports the accuracy, precision, recall and F1 scores of all seven labeling functions that we used in our solution, both types of generative models (Majority vote and the probabilistic graphical model) and the supervised discriminative classifier.

We take the accuracy score of OpineDB pairing method directly from their paper [31] as we use the same test set. However, they do not report their precision, recall and F1 scores. In Table 5, lf_bert_**l:h** corresponds to the labeling function that is based on BERT using the attention head number **h** at layer **l**. lf_tree_op is the labeling function that uses the parse tree and that goes from each opinion to its closest aspect. lf_tree_as travels from aspects to opinions. We train the model with Booking.com dataset for hotels.

SACCS 's pairing model outperforms that of [31] by a margin of 3.03% in accuracy. This result confirms the effectiveness of data programming and weak supervision, and shows that really robust and efficient deep learning models can be designed with little effort and much less resources rather than relying on costly manual annotation. In Table 5, the labeling functions have different accuracies but they all suffer from low recall. We believe this phenomenon is due to the fact that labeling functions are simple heuristics in the first place, and thus fail to cover the entirety of the input space. On the other hand, they all enjoy very high precision, ranging from 91.00% to 95.02%. This insight sheds some light on the nature of our labeling functions and suggests to direct future work on designing heuristics that are less-precise but wide-reaching in order to balance precision and recall ratios. The generative models in Table 5 inherit the high precision of the labeling functions, with the probabilistic model scoring an outstanding 98.10%. However, they also drag the low recall, but are better in general than labeling functions when taken separately. This is due to the nature of generative models which maximize

the benefits of labeling functions while minimizing their risk by combining and integrating their respective labels. Our findings support the original statements of [48]. Nevertheless, the experiment shows that the majority vote model surpasses the probabilistic graphical model in terms of accuracy, unlike what [48] reported. One explanation for this is that our labeling functions are already accurate enough and have comparable F1 scores, leading to similar votes. Thus, consensus should be relatively easier to reach, translating to better accuracy. Finally, we find that the discriminative model is the top scoring model in both accuracy, recall and F1, because it has been trained in a supervised fashion. Rather than depending on labeling functions to provide noisy labels, the discriminative model analyzes the feature space and generalizes its classification decisions to new and potentially unseen input; hence the high recall.

7 CONCLUSION

We proposed *SACCS* : a Natural Language Understanding module for task-oriented dialog systems which allows to recognize the subjective signals in user utterances and filter search results accordingly. *SACCS* is based on the inverted index data structure and mines subjective information from online reviews. We propose a novel subjective tag extraction pipeline that is robust against variations of natural language. We also propose two novel heuristics for pairing an aspect to an opinion. These heuristics aim to overcome the limitation of word-based distance approaches for pairing an aspect term to an opinion term. We also performed extensive evaluation of the proposed subjective tag extraction and pairing techniques. The performed experiments show that these techniques outperform existing approaches.

The advancements brought by *SACCS* are promising, albeit far from perfect. As future work, we plan to investigate the incorporation of search automata as a substitute for inverted indexes. Subjective digital assistants should be able to take into account user profiles and adjust their search and interaction behavior accordingly. We also plan to extend the robustness of the proposed techniques to cater for biased or fraudulent online reviews. For instance, a reviewer might have been paid by a business owner to write positive reviews about it, or negative reviews about its competitors. We have to differentiate between truthful and fake reviews in order to provide a transparent search experience for users. Finally, given the importance of thresholds in similarity assessments, it would be useful for *SACCS* to adjust these dynamically depending on the semantics of the subjective tags being compared.

REFERENCES

[1] 2020. *MS Windows NT The Yelp Dataset*. https://www.yelp.com/dataset/
[2] Stephen H Bach, Bryan He, Alexander Ratner, and Christopher Ré. 2017. Learning the structure of generative models without labeled data. *Proceedings of machine learning research* 70 (2017), 273.
[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).
[4] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. 2017. A survey on dialogue systems: Recent advances and new frontiers. *Acm Sigkdd Explorations Newsletter* 19, 2 (2017), 25–35.
[5] D Manning Christopher, Raghavan Prabhakar, and Schacetzel Hinrich. 2008. Introduction to information retrieval. *An Introduction To Information Retrieval* 151, 177 (2008), 5.

[6] John M Danskin. 2012. *The theory of max-min and its application to weapons allocation problems.* Vol. 5. Springer Science & Business Media.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[8] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075* (2015).

[9] Manaal Faruqui, Jesse Dodge, Sujay K Jauhar, Chris Dyer, Eduard Hovy, and Noah A Smith. 2014. Retrofitting word vectors to semantic lexicons. *arXiv preprint arXiv:1411.4166* (2014).

[10] G David Forney. 1973. The viterbi algorithm. *Proc. IEEE* 61, 3 (1973), 268–278.

[11] Kavita Ganesan and Chengxiang Zhai. 2012. Opinion-based entity ranking. *Information retrieval* 15, 2 (2012), 116–150.

[12] Jianfeng Gao, Michel Galley, Lihong Li, et al. 2019. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval* 13, 2-3 (2019), 127–298.

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[14] Alon Y Halevy. 2019. The Ubiquity of Subjectivity. *IEEE Data Eng. Bull.* 42, 1 (2019), 6–9.

[15] Homa B Hashemi, Amir Asiaee, and Reiner Kraft. 2016. Query intent detection using convolutional neural networks. In *International Conference on Web Search and Data Mining, Workshop on Query Understanding.*

[16] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[17] Dichao Hu. 2019. An introductory survey on attention mechanisms in NLP problems. In *Proceedings of SAI Intelligent Systems Conference.* Springer, 432–448.

[18] Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.* 168–177.

[19] Minqing Hu and Bing Liu. 2004. Mining opinion features in customer reviews. In *AAAI*, Vol. 4. 755–760.

[20] K Indhuja and Raj PC Reghu. 2014. Fuzzy logic based sentiment analysis of product review documents. In *2014 First International Conference on Computational Systems and Communications (ICCSC).* IEEE, 18–22.

[21] Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. 2015. Ontologically grounded multi-sense representation learning for semantic vector space models. In *proceedings of the 2015 conference of the north American chapter of the Association for Computational Linguistics: human language technologies.* 683–693.

[22] Wei Jin, Hung Hay Ho, and Rohini K Srihari. 2009. A novel lexicalized HMM-based learning framework for web opinion mining. In *Proceedings of the 26th annual international conference on machine learning*, Vol. 10. Citeseer.

[23] Joo-Kyung Kim, Gokhan Tur, Asli Celikyilmaz, Bin Cao, and Ye-Yi Wang. 2016. Intent detection using semantically enriched word embeddings. In *2016 IEEE Spoken Language Technology Workshop (SLT).* IEEE, 414–419.

[24] Dan Klein and Christopher D Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st annual meeting of the association for computational linguistics.* 423–430.

[25] Dan Klein and Christopher D Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04).* 478–485.

[26] George Klir and Bo Yuan. 1995. *Fuzzy sets and fuzzy logic.* Vol. 4. Prentice hall New Jersey.

[27] Ari Kobren, Pablo Barrio, Oksana Yakhnenko, Johann Hibschman, and Ian Langmore. 2019. Constructing High Precision Knowledge Bases with Subjective and Factual Attributes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* 2050–2058.

[28] John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[29] Parisa Lak and Ozgur Turetken. 2014. Star ratings versus sentiment analysis–a comparison of explicit and implicit measures of opinions. In *2014 47th Hawaii International Conference on System Sciences.* IEEE, 796–805.

[30] Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. 2010. Structure-aware review mining and summarization. In *Proceedings of the 23rd international conference on computational linguistics.* Association for Computational Linguistics, 653–661.

[31] Yuliang Li, Aaron Feng, Jinfeng Li, Saran Mumick, Alon Halevy, Vivian Li, and Wang-Chiew Tan. 2019. Subjective databases. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1330–1343.

[32] Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5, 1 (2012), 1–167.

[33] Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing.* 1433–1443.

[34] Edward Loper and Steven Bird. 2002. NLTK: the natural language toolkit. *arXiv preprint cs/0205028* (2002).

[35] Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. *arXiv preprint arXiv:1603.01354* (2016).

[36] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval.* Cambridge university press.

[37] Russell Mannion, Huw Davies, and Martin Marshall. 2005. Impact of star performance ratings in English acute hospital trusts. *Journal of Health Services Research & Policy* 10, 1 (2005), 18–24.

[38] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2016. Adversarial training methods for semi-supervised text classification. *arXiv preprint arXiv:1605.07725* (2016).

[39] Luiz Rodrigo Cunha Moura, Gustavo Quiroga Souki, et al. 2017. Choosing a Restaurant: Important attributes and related features of a consumer's decision making process. *Revista Turismo em Análise* 28, 2 (2017), 224–244.

[40] Nikola Mrkšić, Ivan Vulić, Diarmuid Ó Séaghdha, Ira Leviant, Roi Reichart, Milica Gašić, Anna Korhonen, and Steve Young. 2017. Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints. *Transactions of the association for Computational Linguistics* 5 (2017), 309–324.

[41] Joakim Nivre and Mario Scholz. 2004. Deterministic dependency parsing of English text. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics.* 64–70.

[42] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[43] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[44] Maria Pontiki, Dimitrios Galanis, Harris Papageorgiou, Suresh Manandhar, and Ion Androutsopoulos. 2015. Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015).* 486–495.

[45] Maria Pontiki, Dimitris Galanis, John Pavlopoulos, Harris Papageorgiou, Ion Androutsopoulos, and Suresh Manandhar. 2014. SemEval-2014 Task 4: Aspect Based Sentiment Analysis. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014).* Association for Computational Linguistics, Dublin, Ireland, 27–35. https://doi.org/10.3115/v1/S14-2004

[46] Chen Qu, Liu Yang, W Bruce Croft, Falk Scholer, and Yongfeng Zhang. 2019. Answer interaction in non-factoid question answering systems. In *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval.* 249–253.

[47] Lance A Ramshaw and Mitchell P Marcus. 1999. Text chunking using transformation-based learning. In *Natural language processing using very large corpora.* Springer, 157–176.

[48] Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid Training Data Creation with Weak Supervision. *Proc. VLDB Endow.* 11, 3 (Nov. 2017), 269–282. https://doi.org/10.14778/3157794.3157797

[49] Alexander J Ratner, Christopher M De Sa, Sen Wu, Daniel Selsam, and Christopher Ré. 2016. Data programming: Creating large training sets, quickly. In *Advances in neural information processing systems.* 3567–3575.

[50] Régis Saint-Paul, Guillaume Raschia, and Noureddine Mouaddib. 2005. General purpose database summarization. In *Proceedings of the 31st international conference on Very large data bases.* 733–744.

[51] Erik F Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050* (2003).

[52] Dilip Kumar Sharma, Rajendra Pamula, and DS Chauhan. 2020. A contemporary combined approach for query expansion. *Multimedia Tools and Applications* (2020), 1–27.

[53] Junfeng Tian, Zhiheng Zhou, Man Lan, and Yuanbin Wu. 2017. Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017).* 191–197.

[54] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems.* 5998–6008.

[55] Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2016. Recursive neural conditional random fields for aspect-based sentiment analysis. *arXiv preprint arXiv:1603.06679* (2016).

[56] Wenya Wang, Sinno Jialin Pan, Daniel Dahlmeier, and Xiaokui Xiao. 2017. Coupled multi-layer attentions for co-extraction of aspect and opinion terms. In *Thirty-First AAAI Conference on Artificial Intelligence.*

[57] Thomas Wolf, L Debut, V Sanh, J Chaumond, C Delangue, A Moi, P Cistac, T Rault, R Louf, M Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv, abs/1910.03771* (2019).

[58] Hu Xu, Bing Liu, Lei Shu, and Philip S Yu. 2019. Bert post-training for review reading comprehension and aspect-based sentiment analysis. *arXiv preprint arXiv:1904.02232* (2019).

[59] Qiang Ye, Rob Law, and Bin Gu. 2009. The impact of online user reviews on hotel room sales. *International Journal of Hospitality Management* 28, 1 (2009), 180–182.

[60] Lotfi A Zadeh. 1975. The concept of a linguistic variable and its application to approximate reasoning—I. *Information sciences* 8, 3 (1975), 199–249.