ADAPT_TCD: An Ontology-Based Context Aware Approach for Contextual Suggestion

Mostafa Bayomi Séamus Lawless

ADAPT Centre, Knowledge and Data Engineering Group,
School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland

bayomim@tcd.ie seamus.lawless@scss.tcd.ie

Abstract. In this paper we give an overview of the participation of the ADAPT Centre, Trinity College Dublin, Ireland, in both phases of the TREC 2016 Contextual Suggestion Track. We present our ontology-based approach that consists of three models that are based on an ontology that was extracted from the Foursquare category hierarchy. The three models are: User Model, Document Model and Rule Model. In the User Model we build two models, one for each phase of the task, based upon the attractions that were rated in the user's profile. The Document Model enriches documents with extra metadata from Foursquare and categories (concepts) from the ontology are attached to each document. The Rule model is used to tune the score for candidate suggestions based on how the context of the trip aligns with the rules in the model. The results of our submitted runs, in both phases, demonstrate the effectiveness of the proposed methods.

1 Introduction

In TREC 2016, we participate in both phases of the Contextual Suggestion track[1]. The track goal is to provide a venue for the evaluation of systems that are able to make suggestions for a particular person (based upon their profile) in a particular context [2]. This year the track consists of two phases. In Phase 1 we were provided with 495 user profiles. Each profile consists of attractions and the user's opinion regarding them. A user could rate an attraction from 0 (strongly uninterested) to 4 (strongly interested). Items are labelled as -1 if the user didn't provide a rating. Additionally, a user's profile has information about the context of the trip that they are about to take. Information such as the *Group* they are travelling with (alone, friends, family, other), the trip *Season* (summer, winter, autumn, spring), the trip *Type* (holiday, business, other), and the trip *Duration* (night out, day trip, weekend, longer).

This year the organisers released a fixed set of candidate suggestions for 272 contexts, each context representing a city in the United States. The candidate suggestions set consists of approximately 1.2 million candidate attractions.

[1] https://sites.google.com/site/treccontext/trec-2016

The first phase of the task required participants to provide a ranked list of up to 50 suggested attractions for the user to visit, from the provided set of attractions, tailored to each user based on his/her profile. In Phase 1, unlike previous years, we were not asked to setup a live server that could listen to requests from users, rather, we were provided with profiles and the suggestion process was to be completed offline.

In Phase 2 (the batch phase), for each provided user profile and contextual preferences, we were asked to rank sets of candidate attractions that had been suggested during the first phase.

In the following we describe our proposed approach for both Phase 1 and Phase 2, and then discuss our submitted runs along with the results achieved.

2 Proposed Approach

Our approach is based on three models: User Model, Document Model and Rule Model. The three models are based on an ontology that was built from the Foursquare category hierarchy[2].

The ontology is considered the central point of our approach. In the Document Model we enrich the provided documents and identify the set of classes from the ontology that a document belongs to (see Section 2.2). The Rule Model consists of context-specific rules, and penalises documents that are instances of classes that don't match these rules (see Section 2.3). In the User Model we suggest and rank the candidate documents based on their semantic and tag similarity scores with the user model (Sections 3 & 4).

2.1 Building The Ontology

An ontology is a formal naming and definition of the types, properties, and interrelationships of the entities that exist for a particular domain. It describes individuals (instances), classes (concepts), attributes, and relations. In our approach we used an ontology to model the users and the attractions. We exploited the Foursquare category hierarchy to build this ontology.

The Foursquare category hierarchy is a hierarchy that represents Foursquare categories and the relations between them. For example, "*American Restaurant*" is a category in the hierarchy that is a child of the "*Food"* category. Each attraction in Foursquare belongs to one or more of these categories. Hence, we considered a category as a class in the ontology and the attractions that belong to that category as instances of that class.

---

[2] https://developer.foursquare.com/categorytree

2.2 Document Model

The first step in our approach is to model documents (attractions) and enrich them. The organisers released a set of approximately 1.2 million candidate suggestions situated across 272 context cities. Each candidate attraction has an attraction ID, a context city ID, a title, and a URL.

Some of the provided attractions' URLs are the attraction's home page. The home page of an attraction is expected to contain content that is highly relevant to the attraction. This means that the content of the web page should only be related to what this attraction is, or what this attraction serves (in the case where the attraction is a restaurant), and this page should not contain any information about users' ratings or opinions about that attraction. Hence, we leveraged Foursquare to enrich the attractions and model them.

For each attraction, we started by querying its title and location against the Foursquare website[3]. The Foursquare search returns a list of attractions that are sorted based on their relevance to the query. We select the first (most relevant) attraction from that list.

In some cases, the selected attraction is not the actual attraction that we are looking for. This is usually due to one of three reasons:

1- The attraction that we are looking for is not in Foursquare and thus Foursquare search returns any available attractions for the location that was given in the query.

2- The original title of the attraction in the TREC dataset, which we used to generate a query, is not formatted in a manner suited for use as a query on Foursquare. For example, the title is too long or is the description of the attraction.

3- The original title of the attraction in the TREC dataset is not correct for the attraction. For example, the attraction could be a profile page for someone on Facebook and the title is for a coffee house.

Hence, we apply a filter process to check if the attraction returned from Foursquare is, in fact, the attraction that we are looking for. We measure the lexical overlap between the attraction's original title (from the TREC dataset) and the returned attraction's title (from Foursquare). If the lexical overlap score between the two titles exceeds a predefined threshold, the returned attraction is considered to be a correct match and we use it in the next step. If the overlap score is under the threshold, we ignore that attraction and do not include it in the final dataset. The threshold was set based on the length of the attraction's original title after removing common words.

After retrieving the attraction's page from Foursquare we start to parse and extract valuable information such as:

---

[3] https://foursquare.com/

a) List of categories (e.g. *American Restaurant, Park, Museum*).
b) Average users' rating.
c) List of tags[4] (e.g. "*good for a quick meal", "clean", "good for families"*).
d) Users' rating count (number of users who rated this attraction).
e) Users' review count (number of users who gave a review for this attraction).

The list of categories identifies to which category (or set of categories) an attraction belongs. As we mentioned before, we consider a category as a class in our ontology. This class will be used to measure the semantic similarity between an attraction and the user model (see Section 4.2).

We extract users' rating and review counts to be used in the inter-ranking process between attractions. The intuition behind using those counts is: if an attraction *a* has been rated by 500 users and attraction *b* has been rated by 3 users, this means that *a* is more credible than *b* even if *b* has a rate higher than *a*. The same applies for the review count as well.

2.3 Rule Model

The Rule Model consists of rules that identify the relevance of an attraction to the context of the trip using the class (or classes) that the attraction is an instance of. The trip context comprises: the *Group* that a user is travelling with (alone, friends, family, other), the trip *Season* (summer, winter, autumn, spring), the trip *Type* (holiday, business, other), and the trip *Duration* (night out, day trip, weekend, longer). Each of these properties has a set of classes in the ontology that are not suitable for that property. For example, the "*beach*" class is of less relevance when a trip has a *Season* of "winter".

If at least one of the classes that the attraction is an instance of violates at least one rule in the model, then this attraction is ignored (in Phase 1) or given a lower rank (in Phase 2). For example, consider a user for whom the "*Museum*" class is listed as one of their favourite classes in the user model. This user's trip has a *Duration* of "*night out*". Since a museum is typically not suitable for a night out, in the rule model the "*Museum*" category is incompatible with the "*night out*" trip *Duration*. Hence, all attractions that are instances of "*Museum*" are ignored (or given lower rank) and are not selected by our approach even if the user has a preference for museums in her/his user model.

[4] Foursquare tags are keywords extracted from users' reviews

3 Phase 1

3.1 User Modelling

In Phase 1, we model the users based on the positively rated attractions in the user's profile. A different strategy was taken in Phase 2 (see section 4.1). The positively rated attractions are those that have got a rating of either 3 or 4 by the user. The user model was built as follows:

1- For each user and for each attraction we create an index of all the classes (e.g. *American Restaurant*), based on Foursquare data, that the user has rated positively along with the tag set t_c that were found on that attraction's page on Foursquare.
2- We compute the count per class and merge all the tags into one list. The tags are then added to the positive user model.
3- By having all positive classes, we compute the percentage of each class in the positive model. For example, "*American Restaurant*" class represents 31% of the classes in the positive model.

Figure 1 shows a sample of a user's positive model.

For a given place *p* where a user is travelling to, we start to select the documents that match the classes in the positive model. Then we eliminate the documents that belong to a class that violates at least one rule in the rule model. As the target for this phase is to return a list of up to 50 ranked suggestions, we mapped the class percentage in the user model to 50 and represented it as a number, say *x*. Then we start to select the top *x* attractions of this class from the retrieved documents after ranking them.

Ranking between documents of the same class is achieved based on four features:

1- The average users' rating.
2- The users' rating count.
3- The users' review count.
4- The tag similarity measure between a document's tags set t_d and the classes' tags set t_c (see Section 4.3).

For example, for a given class *c*, say "*American Restaurant*", in the user model, we select all documents that are instances of this class (D_c) and eliminate documents that are also instances of classes that violate rules in the rule model (D_c'). Then we rank the remaining documents based on the aforementioned features. Suppose that class makes up 32% of the classes in the positive user model - 32% of 50 is 16. Hence, we select the top 16 documents of the final set.

After we select all documents for all classes in the user model, according to their percentage, we start to rank the documents based on the first three of the features mentioned before and return the final ranked list. The notion behind not including the tags similarity measure in ranking the documents in the final list is that these documents are not instances of the same class (category), which means that they have different sets of tags. For example, documents that are instances of class "*Gym*" have completely different tags than documents that are instances of class "*Japanese Restaurant*".

```
{    "id" : 1,
     .
     "positive_model" : [
          {
               "class" : "American Restaurant",
               "count" : 6,
               "percentage" : 31,
               "tagCloud" : [
                              "ghost bar",
                              "shrimp",
                              "good for dates",
                                .
                                .
                                ]
          },
          {
            "class" : "Art Museum",
           "count" : 3,
          "percentage" : 15,
          tagCloud" : [
                         "museums",
                          "free admission",
                            .
                  ]
     }
```


Figure 1. A sample of Phase 1 positive user model

3.2 Not Enough Attractions

For some profiles and for some places, it is possible that the number of attractions belonging to a specific class, in a specific city do not meet the required number. For example, for a specific profile and a specific city, suppose that the required number of attractions that are instances of the "*American Restaurant*" class is 16. However, in that city, there are only 10 attractions with type "*American Restaurant*". In this instance, we supplement the list to make up for the shortfall using two different approaches (for two different runs):

a) For the first run, we compensate for the shortfall by getting more attractions from the other classes in the user model. For example, the "*American Restaurant*" class has only 10 attractions in the city and the required number is 16. The other classes in the user model e.g. "*Museum, Chinese Restaurant, etc*." have more attractions than what are required by the user model. We combine the remaining, overflow attractions from the other classes into a list. We then rank them based on the aforementioned criteria. Finally we then source the missing 6 attractions from the top of the ranked overflow list.
b) For the second run, we compensate for the shortfall for a specific class by looking, in the ontology, for the class that has the highest ontological similarity

to that class (see Section 4.2). For example, the "*New American Restaurant*" class is considered the highest ontologically similar class to the "*American Restaurant*" class. We then retrieve the attractions that belong to that class, rank them and select the number that we require.

4 Phase 2

In Phase 2, we used the same Document Model (Section 2.2) and Rule Model (Section 2.3) but we followed a different strategy to build the user model. The following sub-sections demonstrate how the Phase 2 user model was built.

4.1 User Modelling

For a given user (U), we build two user models: a positive model (U_{pos}) and a negative model (U_{neg}). The positive model is built based on the positively rated attractions in the user's profile (attractions that were given a rating of either 3 or 4 by the user) and the negative model is built based on the negatively rated attractions in the user's profile (attractions that were given a rating of either 0 or 1 by the user).

In this phase, we do not compute the count per class, instead, we just collect the documents' classes and their tags in each model. Figure 2 illustrates what the user model looks like in Phase 2.

Both user models (positive and negative) may have classes in common, i.e. the user could have positively rated an attraction that is an instance of "*American Restaurant*" and have negatively rated another attraction that is also an instance of "*American Restaurant*". This is why we harvest the tag list from Foursquare. This list of tags is considered an indication of why the user has given a positive or negative rating for that attraction. Furthermore, the negative model could have some classes that are not in the positive model, which in turn means that these classes are not preferable for that user. Hence, we add these classes to the rule model to give a lower rank to documents that are instances of these classes along with the classes that are already in the rule model.

After modelling the user profile and modelling the candidate suggestions, we divide these candidate suggestions into three groups:

Group 1: Working URLs and do not violate rules.
Group 2: Working URLs and violate rules.
Group 3: Not working URLs.

While checking the URLs of the candidate suggestions, it was noted that some were not working, hence, we put these documents, for each profile, in one group (Group 3) and move this group to the end of the final ranked list. Group 1 contains the documents that have working URLs and their classes do not violate any rule in the rule model while Group 2 contains the documents that have working URLs but their classes violate at least one of the rules in the rule model.

```
{   "id" : 1,
    .
    .
    "positive_model" : [
        {
            "doc_id" : "TRECCS-00007296-382",
            "classes" : ["Burrito Place", "Mexican Restaurant"],
            "tagCloud" : [ "burritos",
                           "good for a quick meal",
                           "salsa roja",
                             .
                             .
                             ]
        },
    "negative_model" : [
        {
            "doc_id" : "TRECCS-00007296-382",
            "classes" : "Café",
          "tagCloud" : [ "oatmeal",
                         "banana nut"
                         "corned beef",
                             .
                             .
                    ]
        }
    ]
    .
```


Figure 2. A sample of Phase 2 user models

4.2 Ontology-Based Semantic Similarity

Our approach is based upon the ontological similarity between a document and all the documents in the user's positive profile U_{pos}. i.e. for a given document *d*, the semantic similarity between that document and the user's positive model is the summation of weighted semantic similarity scores between the document's class set and the class set of every document in the positive model.

Semantic similarity has been widely used in many research fields such as Information Retrieval [3] and Text Segmentation [1]. Ontology-based similarity can be classified into three main approaches: Edge-counting, Feature-based and Information Content (IC) based approaches. In our approach, we rely on an Edge-counting approach proposed by Wu and Palmer [5] as its performance is deemed better than other methods [4].

The principle behind Wu and Palmer's similarity computation is based on the edge-counting method, whereby the similarity of two concepts is defined by how closely they are related in the hierarchy, i.e., their structural relations. Given two concepts *c1* and *c2*, the conceptual similarity between them is:

$$ConSim\ (c1, c2) = 2*N / (N1+N2) \qquad (1)$$

Where N is the distance between the closest common ancestor (CS) of *c1* and *c2* and the hierarchy root, and *N1* and *N2* are the distances between the hierarchy root on one hand and *c1* and *c2* on the other hand respectively.

The similarity between two documents can be defined as a summation of weighted similarities between pairs of classes in each of the documents. Given two documents *d1* and *d2*, where *d1* is a document in the candidate suggestions set and *d2* is a document in the user's positive model, the similarity between the two documents is:

$$docSim(d1, d2) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} ConSim(c_i, c_j)}{m \times n} \qquad (2)$$

Where *m* and *n* are the two sets of classes that *d1* and *d2* have respectively.

As mentioned above, the semantic similarity between one of the candidate suggestion documents and the user's positive model can be defined as a summation of weighted semantic similarity scores between the document's class set and the class set of every document in the positive model:

$$semanticSim(d, U_{pos}) = \sum_{i=1}^{D} docSim(d, di) \qquad (3)$$

Where *D* is the set of documents in the positive model.

4.3 Tags similarity

The list of tags associated with an attraction is the list of keywords extracted from the users' reviews about that attraction on Foursquare. These keywords (tags) are considered a representation of the "*taste*" of the attraction and depicts why a user has rated this attraction positively or negatively. For example, consider a user who has positively rated an attraction that has a list of tags that contains: "*sushi bar, good for dates"*. This means that a candidate suggestion that has similar tags is deemed to be more preferable by the user. In our approach we use tag similarity as a factor in the ranking process. We measure the lexical (word) overlap between an attraction's tags list and the documents' tags list in the user's positive model and add the score to the final ranking score.

Consider *d1* is a document in the candidate suggestion set and *d2* is a document in the user's positive model, the similarity between the two documents is:

$$tSim(d1, d2) = \frac{No.of\ overlapped\ words}{x \times y} \qquad (4)$$

Where *x* and *y* are the two sets of tags that *d1* and *d2* have respectively.

The total tag similarity score of a candidate suggestion is:

$$d_{tagsSim} = \sum_{i=1}^{D} tSim(d, di) \quad (5)$$

Where *D* is the set of documents in the positive model.

4.4 Ranking Methodology

For each candidate suggestion in the user profile, we measure two similarity scores:

1- Semantic Similarity (*semanticSim*)
2- Tag Similarity (*tagSim*).

The final ranking score is calculated as follows:

$$d_{totalSim} = d_{semanticSim} + d_{tagSim} + d_{rating} \quad (6)$$

Where d_{rating} is the document's rating on Foursquare.

As mentioned above, we divided the candidate suggestions into three main groups:

1- Working URLs and do not violate rules.
2- Working URLs and violate rules.
3- Not working URLs.

We applied our ranking methodology separately for each group and then merged the groups, after ranking, into the afore-mentioned order to create the final list.

5 Results

We participated in the two phases of the Contextual Suggestion track. In Phase 1, we submitted two runs (see Section 3.2) and in Phase 2 we submitted 3 runs.

This year, the Track organisers evaluated all submitted runs using three evaluation metrics, namely, Reciprocal Rank, P@5 and ndcg@5.

5.1 Phase 1 Results

In this phase we submitted two runs:

1- ADAPT_TCD_r1 (Section 3.2.a)
2- ADAPT_TCD_r2 (Section 3.2.a)

Table 1 reports the performance of our two submitted runs together with the TREC Median

Runs	Reciprocal Rank	P@5	ndcg@5
ADAPT_TCD_r1	**0.5777**	**0.4066**	**0.2643**
ADAPT_TCD_r2	**0.5512**	**0.4098**	**0.2595**
TREC Median	0.5041	0.3508	0.2133

Table 1. Results of our runs in Phase 1

As the results show, our runs are competitive and perform above the TREC median in both cases. In particular, the first run is the best of the two. Overall, the results for both of our runs exhibit promising performance.

5.2 Phase 2 Results

In this phase we submitted three runs:

1- ADAPT_TCD_br1: in this run, we measure the similarity between the candidate suggestions and the user positive model based on the semantic and tag similarity measures (see Section 4.4).

2- ADAPT_TCD_br2: in this run we divided the positive user model into two positive models: positive_4 contains the documents that received a rating of 4 from the user. While positive_3 contains the documents with a rating of 3. The similarity between every candidate suggestion and each positive profile attraction is measured. If the score of a candidate suggestion with positive_4 is higher than with positive_3, it is placed in a group that is at the top of the final ranked list.

3- ADAPT_TCD_br3: in this run we followed the same approach as the first run but we did not measure the tag similarity between documents (see Section 4.3). The intuition behind this run is to assess the impact of tags on measuring the relatedness of a candidate suggestion to a user model.

Table 2 reports the performance of our submitted runs in Phase 2 along with the TREC median.

Runs	Reciprocal Rank	P@5	ndcg@5
ADAPT_TCD_br1	0.5472	**0.4241**	**0.2720**
ADAPT_TCD_br2	0.5472	**0.4241**	**0.2720**
ADAPT_TCD_br3	0.5996	0.3931	**0.2612**
TREC Median	**0.6015**	0.3931	0.2562

Table 2. Results of our runs in Phase 2

The results show that the performance of our first two runs outperform the TREC median in two of the evaluation metrics (P@5 and ndcg@5).

Also the results depict that the second run has the same performance as the first one. Which in turn means that dividing the user positive model into two models does not provide any benefit in the ranking process.

The third run has the lowest performance of the three runs (except in the Reciprocal Rank metric), which means that the use of tag similarity (run1 & run2) has a positive impact on ranking performance.

Overall, the results for the first two runs exhibit promising above-median performances, and hence merit further study in the future.

6 Conclusion and Future Work

This paper describes our participation in both phases of the TREC 2016 Contextual Suggestion track. We proposed an approach that consists of three models that are based on an ontology that was extracted from the Foursquare category hierarchy. The User Model is used to model users based upon their profiles, the Document Model is used to model documents and enrich them, and the Rule Model is used to tune the score for candidate suggestions based upon the context of the trip and how it aligns with the rules in the model. In the first phase we submitted two runs. The results exhibit a good performance where both runs outcome the TREC median. In the second phase we submitted three runs, and results also show that our approach for the second phase is promising.

Moving forward, viable future work may involve:

a) Adding some exceptions to the rule model: There could be an allowance made for instances of a class which violate a rule, but have exceptionally high rankings within that city (or exceptionally high rating and review counts). For instance, a visitor to Sydney is likely to want to go see Bondi Beach, even if it is winter time.
b) Enhancing the user model in phase 1 by following the same strategy that we have followed in phase 2 (Section 4.1).

Acknowledgements.
The ADAPT Centre for Digital Content Technology is funded under the SFI Research Centres Programme (Grant 13/RC/2106) and is co-funded under the European Regional Development Fund.

7 References

[1] Bayomi, M., Levacher, K., Ghorab, M.R., and Lawless, S. OntoSeg: A Novel Approach to Text Segmentation Using Ontological Similarity. *2015 IEEE International Conference on Data Mining Workshop (ICDMW)*, (2015),

[2] Dean-Hall, A., Clarke, C.L., Kamps, J., Thomas, P., Simone, N., and Voorhees, E. *Overview of the TREC 2013 contextual suggestion track*. 2013.

[3] Hliaoutakis, Angelos (Technical University of Crete (TUC), G., Varelas, Giannis (Technical University of Crete (TUC), G., Voutsakis, E., Petrakis, G.M., E., and Milios, E. Information Retrieval by Semantic Similarity. *International Journal on Semantic Web and Information Systems (IJSWIS) 2*, 3 (2006).

[4] Lin, D. An information-theoretic definition of similarity. *ICML*, (1998), 296–304.

[5] Wu, Z. and Palmer, M. Verbs Semantics and Lexical Selection. *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics (1994), 133–138.