# Taxonomy Extraction for Customer Service Knowledge Base Construction

Bianca Pereira[1(✉)], Cecile Robin[1], Tobias Daudert[1], John P. McCrae[1], Pranab Mohanty[2], and Paul Buitelaar[1]

[1] Insight Centre for Data Analytics, Data Science Institute, National University of Ireland Galway, Galway, Ireland
{bianca.pereira,cecile.robin,tobias.daudert,john.mccrae, paul.buitelaar}@insight-centre.org
[2] Fidelity Investments, Boston, USA
pranab.mohanty@fmr.com

**Abstract.** Customer service agents play an important role in bridging the gap between customers' vocabulary and business terms. In a scenario where organisations are moving into semi-automatic customer service, semantic technologies with capacity to bridge this gap become a necessity. In this paper we explore the use of automatic taxonomy extraction from text as a means to reconstruct a customer-agent taxonomic vocabulary. We evaluate our proposed solution in an industry use case scenario in the financial domain and show that our approaches for automated term extraction and using in-domain training for taxonomy construction can improve the quality of automatically constructed taxonomic knowledge bases.

**Keywords:** Taxonomy extraction · Knowledge base construction · Conversational agents

## 1 Introduction

Customer service agents are charged with the role of identifying the intention behind customer questions, retrieving the relevant business information that address those queries, and expressing it in a form that the customer is able to understand. However, the increasing demand for customer services and the necessity to scale up a human workforce are an ongoing challenge for businesses. To address that, organisations are turning to semi-automatic customer service through the use of digital conversational agents (DCAs, also referred to as chatbots) for primary contact with customers, leaving human agents to deal mostly with unusual or more complex customer queries.

Some of the challenges faced by DCAs are: the acquisition of domain knowledge, and knowledge representation that can be audited by the business. Whereas human agents may pass through training programmes to understand the business they are working with, DCAs not only need to acquire knowledge about the


M. Acosta et al. (Eds.): SEMANTiCS 2019, LNCS 11702, pp. 175–190, 2019.
https://doi.org/10.1007/978-3-030-33220-4_13

business but also about the way customers express their informational needs. Further, while it is expected that human agents would be responsible for the provision of the correct information needed by the customer, DCAs cannot be held legally responsible. Instead, businesses need to have means to audit the knowledge used by these DCAs in order to minimise the error in the retrieval of information. Therefore, DCAs need to make use of knowledge representation mechanisms that are applicable to their tasks but also interpretable by humans.

In this paper we propose a solution for the automatic generation of taxonomies from customer service dialogue data. Such solution contributes to the conversational agent use case by taking advantage of existing dialogues between customers and agents as learning data about the domain, and by automatically generating taxonomies as semantic structures for auditing the knowledge used by the conversational agent. The main contributions of this paper are: the comparison of state-of-the-art Term Extraction features, the proposal of Taxonomy Likelihood Scores to measure how likely a tree represents an actual taxonomy, and the applicability of our solution to an anonymised financial customer service use case provided by Fidelity Investments.

## 2 Related Work

DCAs need to have domain-specific knowledge in order to be effective interfaces for human-computer interaction. Most attempts to represent such knowledge have been based on extracting information directly from textual sources, such as online discussion forums [19]. However, it has been identified that a structured form of knowledge can provide a useful intermediate step. Sánchez-Díaz et al. [26] used a logical representation to apply chatbots as intelligent tutors, whereas Semantic Web representations have been used in OntBot [3], through manually designed ontologies, and DBpedia bot [7], by using an open domain knowledge graph. However, none of these solutions is based on automatically generated domain-specific taxonomies. (See Abdul-Kader et al. [1] for a survey.)

The approach we have defined in this paper consists of two steps. First we extract the terms that are most relevant to the domain, a task referred to as automatic term recognition (ATR). Current approaches to this task have employed a varied suite of methods for extracting terms from text based on parts of speech and metrics for assessing 'termhood' [15,29], domain modelling [11], and the composition of multiple metrics in an unsupervised manner [5]. More recently, these methods have been combined into off-the-shelf tools such as ATR4S [7] and JATE [31], and our system is a similar implementation to ATR4S.

The second step organizes these terms into a taxonomy. Although similar to hypernym learning [28], the challenges it proposes are quite different (see [10]). Multiple string and grammar-based methods have been proposed, where baseline systems have used string-based metrics with Hearst-like patterns learned from text [23], while more advanced ones have been based on the concept of *endocentricity* of terms to indicate a hypernym-like relationship [30]. Other methods not based on grammar such as genetic algorithms [14] or word embeddings [17,27]

have also been explored. We include these approaches in our system thus providing an integrated solution to both term and taxonomy extraction.

## 3 Task Definition

Taxonomy extraction consists of extracting a set of terms from a corpus and organising them into a taxonomic structure [10], i.e. a hierarchical tree structure where terms closer to the root of the tree are considered broader[1] than terms farther to the root.

Our framework for taxonomy extraction from text is divided into two steps: automatic term recognition and taxonomy construction. The term recognition step aims at detecting the most relevant terms (also referred to as keyphrases) for a domain represented by a corpus. Based on this list of terms, the taxonomy construction step aims at finding a suitable taxonomy that maintains the correct broad/narrow relationship between terms.

**Definition 1.** *Given a set of terms $T$ in which each $t \in T$ is represented by a label $t_l$. A* ***taxonomy****, $\mathcal{T} = (T, \sqsubseteq)$, is then defined as a partial ordering of a set of terms, $T$, satisfying the following constraints:*

- *(Reflexivity) $t \sqsubseteq t \; \forall t \in T$*
- *(Antisymmetry) $t_1 \sqsubseteq t_2$ and $t_2 \sqsubseteq t_1$ if and only if $t_1 = t_2 \; \forall t_1, t_2 \in T$*
- *(Transitivity) $t_1 \sqsubseteq t_2$ and $t_2 \sqsubseteq t_3$ implies $t_1 \sqsubseteq t_3 \; \forall t_1, t_2, t_3 \in T$*
- *(Unique parent) if $t_1 \sqsubseteq t_2$ and $t_1 \sqsubseteq t_3$ then $t_2 \sqsubseteq t_3$ or $t_3 \sqsubseteq t_2 \; \forall t_1, t_2, t_3 \in T$*
- *(Single root) There is some element $r \in T$ such that $t \sqsubseteq r \; \forall t \in T$*

## 4 Automatic Term Recognition

Our proposed solution for term recognition uses a corpus-based approach based on a pipeline of four consecutive steps (Fig. 1): (i) identification of candidate terms, (ii) scoring, (iii) ranking, and (iv) filtering.

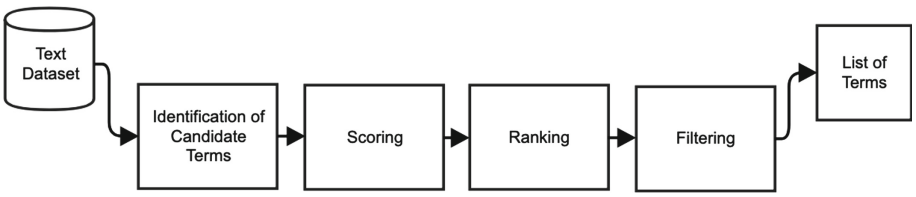


**Fig. 1.** Automatic term recognition pipeline.

The **identification of candidate terms** consists of identifying the key terms appearing within each document. This is accomplished by detecting all noun phrases appearing in the corpus that:

[1] By 'broader' we do not mean hypernymy, but that the topic has a wider scope.

- Contain a minimum and a maximum number of n-grams.
- Do not either start or end with a stopword [20].
- Follows a set of part-of-speech patterns empirically found to be associated with terms [9,20].
- Occurs within the corpus at least a given number of times.

The **scoring** step provides a quantitative measurement for the relevance of each candidate term to the domain in the corpus. As the notion of relevance changes from one application area to another, the scoring step can make use of one or multiple scoring functions more suitable for the underlying domain and task. In this work we will explore the use of multiple combinations of scoring functions for the choice of terms in the financial customer service domain.

Formally, consider $C$ a corpus and $t \in C$ a candidate term extracted in the step for identification of candidate terms. The score for a given term $t$ is a n-tuple $score(t) = (f_1(t), f_2(t), ..., f_n(t))$ given by a set of functions that indicate the relevance of $t$ in $C$ for a task $T$.

In this work we evaluate scoring functions within four categories:

- *Frequency of occurrences*: scoring functions that consider only frequencies of candidate terms within the corpus and/or frequency of words occurring within candidate terms (TF-IDF, Residual IDF [13], C Value [4], ComboBasic [6]).
- *Context of occurrences*: scoring functions that follow the distributional hypothesis [18] to distinguish terms from non-terms by considering the distribution of words in their contexts (PostRankDC [11]).
- *Reference corpora*: scoring functions based on the assumption that terms can be distinguished from other words and collocations by comparing occurrence statistics in the dataset against statistics from a reference corpus - usually of general language/non specific domain (Weirdness [2], Relevance [24]).
- *Topic modelling*: scoring functions based on the idea that topic modelling uncovers semantic information useful for term recognition, in particular that the distribution of words over the topics found by the topic modelling is a less noisy signal than the simple frequency of occurrences (Novel Topic Model (NTM) [21]).

The **ranking** step sorts all candidate terms from the most relevant (i.e. highest score value) to the least relevant (i.e. lowest score value). However, depending on the amount of scoring functions used, the ranking mechanism will be different:

- *Single score*: where only one score function is used, all terms are sorted in ascending order of their associated score value.
- *Voting*: when more than one scoring function is used, the ranking is based on the voting mechanism from [32] and happens in two steps. In the first step, the single score procedure is applied to each scoring function used, resulting in a set of ranked lists R – one list per scoring function. Next, the final ranking position for a candidate term $t$ is given by Eq. 1 where $n$ is the number of scoring functions used and $R_i(t)$ is the rank position of $t$ as provided by the scoring function $i$.

$$rank(t) = \sum_{i}^{n} \frac{1}{R_i(t)} \tag{1}$$

Last, the **filtering** step keeps only the top $n$ terms after the ranking step (where $n$ is a parameter provided to the algorithm).

## 5 Taxonomy Construction

Taxonomy construction aims to build a taxonomy based on the terms extracted by the automatic term recognition algorithm. The proposed pipeline consists of two consecutive steps (Fig. 2): (i) pairwise scoring, and (ii) search. In the first step, each pair of terms extracted $\{\forall(c,d) \in T | c \neq d\}$ will receive a score referring to the estimated probability $p(c \sqsubseteq d)$ that $c$ is a narrower term than $d$. This score will be based on the terms themselves and on their corpus frequency. Based on this set of scores, the second step will search for a tree structure that maximises the likelihood of it being a taxonomy, according to a pre-defined taxonomy likelihood score. The result of this process is a taxonomy $\mathcal{T}$ containing all the terms provided as input.

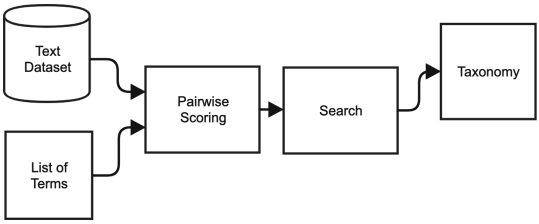


**Fig. 2.** Taxonomy construction pipeline.

### 5.1 Pairwise Scoring

Pairwise scoring aims at identifying, for a pair of terms $(c, d)$ where $c \neq d$, if $c$ is a narrower concept than $d$ ($c \sqsubseteq d$). In this work we accomplished this by using a supervised learning setting.

For each pair of terms $(c, d)$ a feature vector is created with features from the following four categories:

– *String-based features:* Features in this category presume that shorter terms embedded within longer ones are more general. For instance, 'funds' is more general than 'mutual funds'. Features in this category are: Inclusion, overlap, and longest common substring (LCS). Inclusion is $+1$ if $c$ is totally contained within $d$, $-1$ if $d$ is totally contained within $c$, or 0 otherwise. Overlap represents how many words are shared between two terms. Last, LCS measures the longest string of characters shared by two terms.

– *Frequency-based features:* This category assumes that the generality or specificity of a term influences its frequency in the domain corpus. Features in this category are: relative frequency and term difference. Relative frequency measures the difference between the frequency of two terms (Eq. 2 where $f(x)$ denotes the frequency of term $x$). Term difference measures the difference in the distribution of two terms (Eq. 3 where $D(x)$ denotes the number of documents in which the term $x$ appears).

$$relativeFrequency(c,d) = log(\frac{f(d)}{f(c)}) \tag{2}$$

$$termDifference = \frac{|D(c) \cap D(d)|}{|D(d)|} \tag{3}$$

– *Word embedding features:* Features in this category intend to capture the generality of a term by the relation it has with other terms in a shared latent semantic space. For that, GloVe [25] vectors are gathered for each word within a term $c$ in order to generate a vector $v_c$ for the whole term using Single Value Decomposition (SVD) [27]. Two approaches are used, each one leading to a different word embedding feature: SVD average and SVD minmax. The word embedding (WE) features used for pairwise scoring between a pair $(c, d)$ will be calculated according to Eq. 4.

$$WE(c,d) = v_c^T A v_d \tag{4}$$

– *Lexical features:* Features in this category take advantage of existing lexical databases (e.g. Wordnet [22]) with information on the generality of terms (i.e. $c \sqsubseteq d$). Features available are: complete hypernyms (CH) and word hypernyms (WH). The CH feature measures if both terms appear related (directly or indirectly) within the background database, whereas the WH feature measures the presence of relations between any two pairs of words in terms $c$ and $d$ according to Eq. 5.

$$WH(c,d) = \frac{\#words(t_c) + \#words(t_d)}{2} \sum CH(w_c, w_d) \tag{5}$$

A SVM (Support Vector Machine) classifier[2] is then trained using a manually created taxonomy with an associated corpus in the same domain. Each pair of terms $(c, d)$ where $c \neq d$ and $c$ is a child of $d$ in the training taxonomy is labelled with $+1$, otherwise the pair is labelled as $-1$. The result of the classification is the probability estimation for the class $+1$ which is then given as the estimate for $p(c \sqsubseteq d)$.

### 5.2 Search

Based on the pairwise score between any two terms provided as input to the taxonomy construction, the search step aims at identifying a tree structure that

[2] Earlier versions of the system experimented with other classifiers, however we have found that SVMs provide higher quality and robust performance.

represents a taxonomy of these terms. In order to identify how close a tree structure is to a valid taxonomy, a taxonomy likelihood score was designed. In this work we design and evaluate three score functions: transitive, non-transitive, and Bhattacharyya-Poisson. The goal for the search is to find a tree-structure that maximises the taxonomy likelihood score. Here we also experimented with two types of search methods: greedy and beam.

**Taxonomy Likelihood Score**

*Transitive.* The transitive score (Eq. 6) just follows the basic assumption that the best taxonomy is the one that maximises the product of all $p(c \sqsubseteq d)$ for all $(c, d)$ pairs of terms. In practice, we take logs of the probabilities and maximise the resulting sum (Eq. 7).

$$S(T) = \prod_{c \sqsubseteq d} p(c \sqsubseteq d) \tag{6}$$

$$\max_T S(T) = \max_T \sum_{c \sqsubseteq d} \log p(c \sqsubseteq d) \tag{7}$$

*Non-transitive.* In practice, the transitive score is not expected to work well since it is maximised by taxonomies for which there are as few as possible pair of terms $(c, d)$ such that $c \sqsubseteq d$. The most trivial case is when a taxonomy is composed only by a single root term and all other terms are a direct child of it. As such, taxonomies that are constructed from maximising the transitive score tend to have a very large number of average children. In order to avoid that, the non-transitive score (Eq. 8) considers only the set of direct children, which are denoted by $c \leq d$ and should satisfy the following constraints:

- $c \leq d$ implies $c \sqsubseteq d$.
- $c \sqsubseteq d$ implies there exists $e$ such that $c \leq e$ and $e \sqsubseteq d$.
- For all, $c \leq d$ there does not exist $e$, $e \neq d$, $e \neq c$, such that $c \leq e$ and $e \leq d$.

$$S_{nonTransitive}(T) = \prod_{c \leq d} p(c \sqsubseteq d) \tag{8}$$

*Bhattacharyya-Poisson.* Despite the possible improvement given by the non-transitive likelihood function, it may still lead to a single path (i.e. a tree with just one branch), what differs from usual expectations for taxonomies as more balanced trees (i.e. tree structures with multiple branches). In order to address this, the Bhattacharyya-Poisson likelihood score takes into account the number of children of each node in the tree.

Formally, let $n_c$ denote the number of terms in a taxonomy that have exactly $c$ children. If the tree was to be constructed in a truly random fashion we would expect $n_c$ to be distributed according to a binomial distribution (Eq. 9). However, in a completely balanced tree of $N$ terms there are $N - 1$ direct children so that

the number of children in each branch is $p = \frac{(N-1)}{b}$, where $b$ is the number of branches. In order to allow us to vary the average number of children in each branch ($\lambda$), since taxonomies do not need to be completely balanced, we use the Poisson distribution as an approximation for the binomial distribution (Eq. 10). However, the constraints on what constitutes a taxonomy fix this value to very near 1 and we wish to vary this in order to control the taxonomy structure. We thus ignore the leaf nodes in the taxonomy (i.e., we ignore $n_0$).

$$p(n_c = m) = choose(N, m)p^m(1-p)^{N-m} \tag{9}$$

$$p(n_c = m) \simeq \frac{\lambda^m e^{-\lambda}}{m!} \tag{10}$$

In order to measure the similarity of the candidate taxonomy with this theoretical probability value we use the Bhattacharyya distance [8] that measures the similarity of two (discrete) probability distributions $p$ and $q$ as provided in Eq. 11. If we compare this to the actual children count in a taxonomy, we can score a taxonomy as provided in Eq. 12. Finally, this is interpolated with the previous metric to score a taxonomy as provided in Eq. 13.

$$B(p, q) = \sum_i \sqrt{p_i q i} \tag{11}$$

$$BP(T) = \sum_{i=1}^{N} \frac{n_i \lambda^{n_i} e^{-\lambda}}{(N - n_0) n_i!} \tag{12}$$

$$S_{BP}(T) = \prod_{c \leq d} p(c \sqsubseteq d) + \alpha N \times BP(T) \tag{13}$$

**Search Method.** Having chosen the likelihood score, the next step is to use a search strategy that optimises this likelihood score. We used two search strategies: (i) greedy, and (ii) beam.

*Greedy.* In this method, provided the pairwise scores for all pairs of terms, the pair that has the maximal score for $p(c \sqsubseteq d)$ is added as $c \leq d$. This process is repeated until a full taxonomy is constructed, that is we take pairs $c \leq d$ that satisfy the first four axioms of the taxonomy, from the first (reflexivity - unique parent) until the final axiom is satisfied (single root), which means the taxonomy is complete.

*Beam.* In contrast to the greedy method, where only a single partially constructed taxonomy is maintained, this method keeps a list of the top scoring possible solutions. This list (also called beam) is of a fixed size, thus the addition of a new partial solution may cause the least scoring partial solution to be dropped. Complete solutions are stored in a separate beam and the algorithm proceeds until it has considered all possible pairs of concepts and returns the highest scoring complete solution found.

# 6 Experiments

## 6.1 Automatic Term Recognition

Our experiments were performed on an anonymised customer service chat-log dataset containing 300,000 conversations provided by Fidelity Investments, where any personal information was removed prior to using the data for this project. In order to use such dataset, first we need to decide what is the unit of analysis or, in other words, what would be considered a document in the corpus. There are two obvious options: each interaction made by a customer or an agent, or the whole customer-agent conversation. In our experiments we decided to use the whole conversation as unit of analysis for two reasons: (i) to give priority to terms frequent in conversations rather than those cited multiple times only in a single conversation; and (ii) because different elements in a conversation could provide contextual information for the terms extracted.

Also, in order to protect customers' identities and personal data, the corpus provided is full of anonymisation tokens, i.e. tokens that were put in place of what would be sensitive information (e.g. name, email address, etc.). Before conducting any experiment, a list of stopword terms was compiled containing all anonymisation tokens appearing in the corpus so that these would not be captured as terms due to their potential high frequency in the corpus.

After preprocessing the corpus, several experiments were conducted in order to identify the most suitable configuration for automatic term recognition in the customer-agent interaction scenario. First, we adjusted the hyper-parameters for identification of candidate terms: the part-of-speech patterns used are given by the literature [9,20], the list of stopwords includes common English stopwords and the anonymisation tokes extracted, a relevant term should have frequency higher than 2, and it should not have an unlimited size so we choose a maximum of 4-grams and we varied the minimum n-gram between 1 and 2. Table 1 summarises the configuration of each experiment.

For scoring functions we choose TF-IDF as a baseline due to its common use in measuring relevant terms in a corpus. We also opted to have settings with one function from each category to measure how they behave independently, except the reference corpora category since using Wikipedia as background corpus could give a high number of false positive terms if used alone. Also, we included a configuration ($TE_1$ and $TE_2$) that has demonstrated positive results in previous experiments. And last, due to the positive results of the use of ComboBasic ($TE_2$ long) we experimented combining it with a reference corpora scoring function.

**Table 1.** Configuration of each automatic term recognition experiment

| Experiment | n-gram | | Scoring function | | | | Ranking |
|---|---|---|---|---|---|---|---|
| | Min | Max | Frequency | Context | Reference corpora | Topic modelling | |
| TE baseline | 1 | 4 | tf-idf | - | - | - | Single |
| TE baseline long | 2 | 4 | tf-idf | - | - | - | Single |
| $TE_1$ | 1 | 4 | tf-idf<br>Residual Idf<br>C Value<br>ComboBasic | - | Weirdness | - | Voting |
| $TE_1$ long | 2 | 4 | tf-idf<br>Residual Idf<br>C Value<br>ComboBasic | - | Weirdness | - | Voting |
| $TE_2$ | 1 | 4 | ComboBasic | - | - | - | Single |
| $TE_2$ long | 2 | 4 | ComboBasic | - | - | - | Single |
| $TE_3$ long | 2 | 4 | ComboBasic | - | Relevance | - | Voting |
| $TE_4$ | 1 | 4 | - | PostRankDC | - | - | Single |
| $TE_4$ long | 2 | 4 | - | PostRankDC | - | - | Single |
| $TE_5$ | 1 | 4 | - | - | - | NTM | Single |
| $TE_5$ long | 2 | 4 | - | - | - | NTM | Single |

**Results and Discussion.** The evaluation of the automatic term recognition experiments was based on the manual evaluation of a list of terms. A group of domain experts from Fidelity Investments who are familiar with financial industry taxonomies were asked to evaluate the relevance of each term to the financial domain according to a 5-point Likert-type scale (from irrelevant to very relevant). Any term rated as either 4 or 5 by a majority of annotators (i.e. at least four annotators) was considered a relevant term in the financial domain.

The list of terms for evaluation was generated by merging the top 100 terms extracted by each experiment, removing duplicates and ranking them using the Mean Reciprocal Rank [16]. The final list was then limited to a manageable number of terms (200 terms) sent for manual validation by a team of experts in the financial customer service domain. Since each term may appear in more than one experiment, Table 1 reports how much of the evaluation list is covered by each experiment. The result of each experiment is then evaluated using precision, i.e. the proportion of correct terms among those appearing in the evaluation set. Based on this evaluation, the experiments $TE_2$ log and $TE_3$ were the ones to provide the best results in our experiments (Table 2).

The positive results using the ComboBasic feature are mostly due to its ability to remove conversational words (such as "good morning", "thanks"). Because greetings do not appear in the corpus either as part of longer combination of words (e.g. "good morning" will not typically be a substring of any longer noun phrase), or as an aggregation of smaller and frequent terms (e.g. "morning" is not a frequent term in the corpus and "good" is not even considered a term for

**Table 2.** Evaluation of automatic term recognition experiments

| Experiment | Coverage (%) | Precision (%) |
|---|---|---|
| TE baseline | 36 | 48.6 |
| TE baseline long | 38.5 | 59.7 |
| $TE_1$ | 47 | 50.0 |
| $TE_1$ long | 41.5 | 59.0 |
| $TE_2$ | 30.5 | 54.1 |
| $TE_2$ long | 37.5 | **65.3** |
| $TE_3$ long | 37.5 | **65.3** |
| $TE_4$ | 45 | 56.7 |
| $TE_4$ long | 30.5 | 62.3 |
| $TE_5$ | 21.5 | 30.2 |
| $TE_5$ long | 21.5 | 30.2 |

being an adjective), then the requirements for frequency and term embeddedness expected by ComboBasic will be less likely to consider greetings as terms.

The drawback of using ComboBasic, on the other hand, is that it fails to retrieve terms of different lengths. In fact, only the experiment $TE_5$ retrieved a mix of single and multi-word terms. Also, other irrelevant terms that could have been removed by filtering out common out-of-domain terms using the weirdness feature ($TE_3$ long) did not have the expected result. The interpretation we give is that Wikipedia is not the most suitable corpus for this use case. Instead, in future work, we would like to experiment with customer service data in other domains so that we could remove terms that are common to customer service domain in general while keeping those that are specific to customer service in the financial domain. The difficulty lies in the availability of such datasets.

### 6.2 Taxonomy Construction

The objective of our taxonomy construction experiments are twofold: (i) to evaluate the combination of likelihood score and search methods that generate the best taxonomy structure; and (ii) to verify the impact of using an in-domain taxonomy as training data for the pairwise scoring function.

In order to separate the error generated by the automatic term recognition from the results of the taxonomy construction, we did not use the terms extracted previously. Instead, three manually constructed financial taxonomies provided by Fidelity Investments (financial products, financial sectors, and financial topics) were used to inform the terms to be used in each experiment. The products taxonomy was used to train the in-domain model for pairwise scoring while the remaining taxonomies were used as gold standard for evaluation.

First, two pairwise scoring models were trained using LibSVM [12], a library for support vector classification. The first model is trained on an out-of-domain

taxonomy (food domain) and background corpus provided by the TexEval challenge [10]. The second model is trained using the products taxonomy and the pages in the Fidelity.com website as a corpus for extraction of features required by the Pairwise Scoring algorithm. All relationships between terms $(c, d)$ appearing in the training taxonomy are given as positive examples $(+1)$ and any other combination of terms is a negative example $(-1)$. As the negative class sample is very large we perform negative sampling, with a fixed ratio of 5 negative examples to each positive one. The pairwise scoring uses the probability estimation in LibSVM by returning the probability that the class is $+1$ as $p(c \sqsubseteq d)$.

The workflow for comparison of taxonomy construction configurations using the different models is the following:

- *Step1.* The term extraction algorithm was used to extract from the chatlog dataset the frequency of each term in the gold standard $T_{GOLD}$ taxonomy.
- *Step2.* The taxonomy construction algorithm was applied using the term frequencies from step 1, varying the configuration (Table 3) and model to be evaluated.

**Table 3.** Configuration of each taxonomy construction experiment

| | Likelihood score | | |
|---|---|---|---|
| Search | Transitive | Non-transitive | Bhattacharyya-Poisson |
| Greedy | $TA_1$ | $TA_2$ | $TA_3$ |
| Beam | $TA_4$ | $TA_5$ | $TA_6$ |

The unit of analysis for our evaluation is each pair of concepts $x$ and $y$ where there is a relation $x \sqsubseteq y$. Note that transitivity was not taken into consideration, therefore only direct connection between terms was considered. The results of each run were evaluated using link precision as described in Eq. 14, where $T$ is the resulting taxonomy provided by the taxonomy construction algorithm, and $T_{GOLD}$ is the taxonomy provided as the expected result.

$$precision(T, T_{GOLD}) = \frac{|\{(x \sqsubseteq y) \in T \wedge (x \sqsubseteq y) \in T_{GOLD}\}|}{|\{(x \sqsubseteq y) \in T\}|} \tag{14}$$

**Results and Discussion.** The problem of taxonomy construction is very challenging, and previous evaluations such as TexEval [10] have only reported precision as high as 30%. One challenge is that the structure of multiple taxonomies in the same domain may vary considerably, therefore it is difficult to take advantage of the overall structure of one taxonomy when the best structure in another occasion may be completely different. Therefore, due to the multiple challenges in the automatic generation of a taxonomy structure (see [10]), a precision measure of 20% can already be considered as a strong result.

From the perspective of the logical connection between terms in the taxonomy, the best performing setting (Table 4) was the one using the Bhattacharyya-Poisson likelihood score function, greedy search strategy, and the in-domain model (using the products taxonomy as background knowledge). This setting consistently gave better results than all the others on the three taxonomies available. It is important to note, however, that only the sectors and topics taxonomies were used as gold standard since the products taxonomy was the one used as training data. The results using the products taxonomy are displayed only to contrast the impact of using an in-domain taxonomy versus using an out-of-domain one to train the pairwise scoring model. In fact, the results suggest that the choice of likelihood score and search method have a higher contribution to the quality of the final taxonomy than the domain of the taxonomy used to train the pairwise scoring algorithm. Therefore, we infer that the taxonomy construction framework can be successfully applied to other customer service domains where there is no background taxonomy to train the pairwise scoring model.

**Table 4.** Precision (%) of taxonomy construction algorithm

| Search method | Likelihood score function | Out-of-Domain model | | | In-Domain model | | |
|---|---|---|---|---|---|---|---|
| | | Products | Sectors | Topics | Products | Sectors | Topics |
| Greedy | Transitive | 16.86 | 11.25 | 11.41 | 7.28 | 8.75 | 7.38 |
| | Non-Transitive | 17.62 | 12.50 | 11.41 | 21.84 | 15.00 | 13.42 |
| | BP | 17.62 | 12.50 | 12.08 | 22.22 | 16.25 | 13.42 |
| Beam | Transitive | 4.21 | 1.25 | 1.34 | 3.83 | 1.25 | 0.67 |
| | Non-Transitive | 16.86 | 11.25 | 10.74 | 16.48 | 8.75 | 8.72 |
| | BP | 18.01 | 12.50 | 10.74 | 15.33 | 8.75 | 9.40 |

In general, the pairwise scoring model is just one element that impacts on the final taxonomy built from text. In some cases its use provided better results and in some cases not. Overall, the choices of likelihood score function and search strategy had a higher impact on the quality of the final taxonomy than the taxonomy provided for pairwise scoring training. Nonetheless, the use of a domain taxonomy as background knowledge showed between 10% and 25% improvement in the precision when using the non-transitive score or BP functions with the greedy search.

## 7 Conclusion and Future Work

In this paper, we presented a solution for the automatic extraction of taxonomies, motivated by its use by conversational agents, and applied this solution to an anonymised customer service dialogue data provided by Fidelity Investments.

We evaluated multiple methods for automatic term recognition, where ComboBasic was the most suitable term scoring function for the dataset used.

Also, we introduced multiple functions to evaluate the likelihood that a tree structure is a taxonomy and evaluated their efficacy in taxonomy extraction. Furthermore, our results suggest that despite our approach benefiting from in-domain data, it does not require the taxonomy used for training to be in the same domain of the business, which makes our solution applicable to customer service domains where a manually created taxonomy is not available.

In future work, we plan to explore: the use of customer service text datasets in other domains as background knowledge to remove greetings and other out-of-domain terms, the design of likelihood scores that take into consideration the desired graph structure of the taxonomy, and the application of the extracted taxonomy in a conversational agent.

## References

1. Abdul-Kader, S.A., Woods, J.: Survey on chatbot design techniques in speech conversation systems. Int. J. Adv. Comput. Sci. Appl. **6**(7) (2015)
2. Ahmad, K., Gillam, L., Tostevin, L., et al.: University of surrey participation in TREC8: weirdness indexing for logical document extrapolation and retrieval (wilder). In: TREC, pp. 1–8 (1999)
3. Al-Zubaide, H., Issa, A.A.: Ontbot: ontology based chatbot. In: International Symposium on Innovations in Information and Communications Technology, pp. 7–12. IEEE (2011)
4. Ananiadou, S.: A methodology for automatic term recognition. In: COLING 1994, vol. 2: The 15th International Conference on Computational Linguistics. vol. 2 (1994)
5. Astrakhantsev, N.: Automatic term acquisition from domain-specific text collection by using wikipedia. Proc. Inst. Syst. Program. **26**(4), 7–20 (2014)
6. Astrakhantsev, N.: Methods and software for terminology extraction from domain-specific text collection. Ph.D. thesis, Ph. D. thesis, Institute for System Programming of Russian Academy of Sciences (2015)
7. Athreya, R.G., Ngonga Ngomo, A.C., Usbeck, R.: Enhancing community interactions with data-driven chatbots-the DBpedia chatbot. In: Companion of the The Web Conference 2018 on The Web Conference 2018, pp. 143–146. International World Wide Web Conferences Steering Committee (2018)
8. Bhattacharyya, A.: On a measure of divergence between two statistical populations defined by their probability distributions. Bull. Calcutta Math. Soc. **35**, 99–109 (1943)
9. Bordea, G.: Domain adaptive extraction of topical hierarchies for expertise mining. Ph.D. thesis (2013)
10. Bordea, G., Lefever, E., Buitelaar, P.: SemEval-2016 task 13: taxonomy extraction evaluation (TexEval-2). In: Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1081–1091 (2016)
11. Buitelaar, P., Bordea, G., Polajnar, T.: Domain-independent term extraction through domain modelling. In: The 10th International Conference on Terminology and Artificial Intelligence (TIA 2013), Paris, France. 10th International Conference on Terminology and Artificial Intelligence (2013)
12. Chang, C.C., Lin, C.J.: LIBSVM: a library for support vector machines. ACM Trans. intell. Syst. Technol. (TIST) **2**(3), 27 (2011)

13. Church, K.W., Gale, W.A.: Poisson mixtures. Nat. Lang. Eng. **1**(2), 163–190 (1995)
14. Cleuziou, G., Moreno, J.G.: QASSIT at SemEval-2016 Task 13: on the integration of semantic vectors in pretopological spaces for lexical taxonomy acquisition. In: 10th International Workshop on Semantic Evaluation (SemEval-2016), pp. 1315–1319 (2016)
15. Cram, D., Daille, B.: Terminology extraction with term variant detection. In: Proceedings of ACL-2016 System Demonstrations, pp. 13–18 (2016)
16. Craswell, N.: Mean reciprocal rank. In: Encyclopedia of Database Systems, pp. 1703–1703 (2009)
17. Fu, R., Guo, J., Qin, B., Che, W., Wang, H., Liu, T.: Learning semantic hierarchies via word embeddings. In: Proceedings of the 2014 Conference of the Association for Computational Linguistics, pp. 1199–1209 (2014)
18. Harris, Z.S.: Distributional structure. Word **10**(2–3), 146–162 (1954)
19. Huang, J., Zhou, M., Yang, D.: Extracting chatbot knowledge from online discussion forums. IJCAI **7**, 423–428 (2007)
20. Hulth, A.: Enhancing linguistically oriented automatic keyword extraction. In: Proceedings of HLT-NAACL 2004: Short Papers (2004)
21. Li, S., Li, J., Song, T., Li, W., Chang, B.: A novel topic model for automatic term extraction. In: Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 885–888. ACM (2013)
22. Miller, G.: WordNet: An Electronic Lexical Database. MIT Press, Cambridge (1998)
23. Panchenko, A., et al.: TAXI at SemEval-2016 Task 13: a taxonomy induction method based on lexico-syntactic patterns, substrings and focused crawling. In: 10th International Workshop on Semantic Evaluation (SemEval-2016) (2016)
24. Peñas, A., Verdejo, F., Gonzalo, J.: Corpus-based terminology extraction applied to information access. In: Proceedings of Corpus Linguistics. vol. 2001, p. 458. Citeseer, Priceton (2001)
25. Pennington, J., Socher, R., Manning, C.: GloVe: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
26. Sánchez-Díaz, X., Ayala-Bastidas, G., Fonseca-Ortiz, P., Garrido, L.: A knowledge-based methodology for building a conversational chatbot as an intelligent tutor. In: Batyrshin, I., Martínez-Villaseñor, M.L., Ponce Espinosa, H.E. (eds.) MICAI 2018. LNCS (LNAI), vol. 11289, pp. 165–175. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-04497-8_14
27. Sarkar, R., McCrae, J.P., Buitelaar, P.: A supervised approach to taxonomy extraction using word embeddings. In: Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018) (2018)
28. Snow, R., Jurafsky, D., Ng, A.Y.: Learning syntactic patterns for automatic hypernym discovery. In: Advances in Neural Information Processing Systems, pp. 1297–1304 (2005)
29. Spasić, I., Greenwood, M., Preece, A., Francis, N., Elwyn, G.: Flexiterm: a flexible term recognition method. J. Biomed. Semant. **4**(1), 27 (2013)
30. Tan, L., Bond, F., van Genabith, J.: USAAR at SemEval-2016 Task 13: Hyponym endocentricity. In: 10th International Workshop on Semantic Evaluation (SemEval-2016) (2016)

31. Zhang, Z., Gao, J., Ciravegna, F.: Jate 2.0: Java automatic term extraction with apache SOLR. In: Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC-2016) (2016)
32. Zhang, Z., Iria, J., Brewster, C., Ciravegna, F.: A comparative evaluation of term recognition algorithms. In: Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC-2008), vol. 5 (2008)