# Geo-mining: Discovery of Road and Transport Networks Using Directional Patterns

**Dmitry Davidov**
ICNC
The Hebrew University of Jerusalem
dmitry@alice.nc.huji.ac.il

**Ari Rappoport**
Institute of Computer Science
The Hebrew University of Jerusalem
arir@cs.huji.ac.il

## Abstract

One of the most desired information types when planning a trip to some place is the knowledge of transport, roads and geographical connectedness of prominent sites in this place. While some transport companies or repositories make some of this information accessible, it is not easy to find, and the majority of information about uncommon places can only be found in web free text such as blogs and forums. In this paper we present an algorithmic framework which allows an automated acquisition of map-like information from the web, based on surface patterns like "from X to Y". Given a set of locations as initial seeds, we retrieve from the web an extended set of locations and produce a map-like network which connects these locations using transport type edges. We evaluate our framework in several settings, producing meaningful and precise connection sets.

## 1 Introduction

Textual geographical information such as location descriptions, directions, travel guides and transport tables is extensively used by people. Discovering such information automatically can assist NLP applications such as question answering (Santos and Cardoso, 2008), and can be useful for a variety of other applications, including automatic map annotation.

Some textual geographical information can be found in web sites of transport companies, tourist sites and repositories such as Wikipedia. Such sites usually utilize structured information such as machine-readable meta-data, tables, schedule forms or lists, which are relatively convenient for processing. However, automatic utilization of such information is limited. Even on these sites, only a small fraction of the available geographical information is stored in a well-structured and freely accessible form. With the growth of the web, information can be frequently found in 'ordinary' web pages such as forums, travelogues or news. In such sites, information is usually noisy, unstructured and present in the form of free text.

This type of information can be addressed by lexical patterns. Patterns were shown to be very useful in all sorts of lexical acquisition tasks, giving high precision results at relatively low computational costs (Pantel et al., 2004). Pattern-driven search engine queries allow to access such information and gather the required data very efficiently (Davidov et al., 2007).

In this paper we present a framework that given a few seed locations as a specification of a region, discovers additional locations (including alternate location names) and map-like travel paths through this region labeled by transport type labels.

The type of output produced by our framework here differs from that in previous pattern-based studies. Unlike mainstream pattern-based web mining, it does not target some specific two-slot relationship and attempts to extract word tuples for this relationship. Instead, it discovers geographical networks of transport or access connections. Such networks are not unstructured sets of word pairs, but a structured graph with labeled edges.

Our algorithm utilizes variations of the basic pre-defined pattern "[Transport] from Location1 to Location2" which allows location names and connections to be captured starting from the given seed location set. We acquire search engine snippets and extract contexts where location names co-appear. Next we construct a location graph and merge transport edges to identify main transport group types. Finally, we improve the obtained data by reducing transitive connections and identifying key locations.

The obtained location data can be used as a draft for preparation of travel resources and on-demand travel plans. It can also be used for question answering systems and for automated enrichment and verification of existing geographical resources.

We evaluate our framework on three different regions of different scale and type: Annapurna in Nepal, the south Israel area and the Cardiff area in England. In our evaluation we estimated precision and the amount of discovered locations and transport edges, and examined the quality of the obtained map as a whole by visually comparing the overall connectedness of the graph to an actual road or transport map.

## 2 Related Work

In this paper we utilize a pattern-based lexical acquisition framework for the discovery of geographical information. Due to the importance of lexical databases for many NLP tasks, substantial research has been done on direct or indirect automated acquisition of concepts (sets of terms sharing a significant aspect of their meaning) and concept relationships in the form of graphs connecting concepts or terms inside concepts into usually hierarchical or bipartite networks. In the case of geo-mining, concepts can include sets of alternative names for some place, or sets of all locations of the same type (e.g., all countries). Geographical relationships can include nearness of two locations and entity-location relationships such as institute-address, capital-country, tourist site-city etc.

The major differences between relationship acquisition frameworks come from the types and annotation requirements of the supplied input and the basic algorithmic approach used to process this input. A first major algorithmic approach is to represent word contexts as vectors in some space and use distributional measures and automatic clustering in that space. Curran (2002) and Lin (1998) use syntactic features in the vector definition. Caraballo (1999) uses conjunction and appositive annotations in the vector representation.While efforts have been made for improving the computational complexity of these methods (Gorman and Curran, 2006), they remain data and computation intensive.

The second major algorithmic approach is to use lexico-syntactic patterns, which have been shown to produce more accurate results than feature vectors at a lower computational cost on large corpora (Pantel et al., 2004). Most related work deals with discovery of hypernymy (Hearst, 1992; Pantel and Lin, 2002) and synonymy (Widdows and Dorow, 2002; Davidov and Rappoport, 2006). Some studies deal with the discovery of more specific relation sub-types, including inter-verb relations (Chklovski and Pantel, 2004) and semantic relations between nominals (Davidov and Rappoport, 2008). Extensive frameworks were proposed for iterative discovery of pre-specified (e.g., (Riloff and Jones, 1999)) and unspecified (e.g., (Agichtein and Gravano, 2000)) relation types.

Some concepts and relationships examined by seed-based discovery methods were of a geographical nature. For example, (Etzioni et al., 2004) discovered a set of countries and (Davidov et al., 2007) discovered diverse country relationships, including location relationships between a country and its capital and a country and its rivers. As noted in Section 1, the type of output that we produce here is not an unstructured collection of word pairs but a labeled network. As such, our task here is much more complex.

Our study is related to geographical information retrieval (GIR) systems. However, our problem is very far from classic GIR problem settings. In GIR, the goal is to classify or retrieve possibly multilingual documents in response to queries in the form 'theme, spatial relationship, location', e.g., 'mountains near New York' (Purves et al., 2006). Our goal, in contrast, is not document retrieval, but the generation of a structured information resource, a labeled location graph.

Spatial relationships used in natural language tend to be qualitative and descriptive rather than quantitative. The concept of Naive Geography, which reflects the way people think and write about geographical space, is described in (Egenhofer and Shariff, 1995). Later in (Egenhofer and Shariff, 1998) they proposed a way to convert coordinate-based relationships between spatial entities to natural language using terms as 'crosses', 'goes through' or 'runs into'. Such terms can be potentially used in patterns to extract geographical information from text. In this paper we start from a different pattern, 'from ... to', which helps in discovering transport or connectedness relationships between places, e.g., 'bus from X to Y' and 'road from X to Y'.

The majority of geographical data mining

frameworks utilize structured data such as available gazetteers and Wikipedia metadata. Several other studies utilize semi-structured data like Wikipedia links (Overell and Ruger, 2007) or substructures in web pages, including addresses and phone numbers (Borges et al., 2007).

The recent Schockaert et al.( 2008) framework for extraction of topological relationships from the web has some similarities to our study. In both cases the algorithm produces map-like structures using the web. However, there are significant differences. They utilize relatively structured address data on web pages and rely on the order of named entities in address data for extracting containment relationships. They also use co-appearances in addresses (e.g., 'R1 / R2' and 'R1 & R2' as in "Newport & Gabalfa, Cardiff") to deduce location boundaries. This allows them to get high precision data for modern and heavily populated regions like Cardiff, where the majority of offices have available well-formatted web pages.

However, in less populated regions (a major target for tourist information requests), this strategy could be problematic since a major information source about these places would be not local web sites (in which local addresses are likely to be found) but foreign visitor sites, web forums and news. We rely on free text available in all types of web pages, which allows us to capture unstructured information which contains a significant portion of the web-available geographical knowledge.

Our goals are also different from Schockaert et al.( 2008), since we focus on obtaining information based on paths and transport between locations, while in their work the goal is to find a network representing nearness of places rather than their connectivity by means of transport or walking. Nevertheless, in one of our evaluation settings we targeted the area of Cardiff as in (Schockaert et al., 2008). This allowed us to make an indirect comparison of a relevant part of our results to previous work, achieving state-of-the-art performance.

## 3 The Algorithm

As input to our algorithm we are given a seed of a few location names specifiying some geographical region. In this section we describe the algorithm which, given these names, extracts the labeled structure of connections between entities in the desired region. We first use a predefined pattern for recursive extraction of the first set of entities. Then we discover additional patterns from co-appearing location pairs and use them to get more terms. Next, we label and merge the obtained location pairs. Finally, we construct and refine the obtained graph.

### 3.1 Pattern-based discovery with web queries

In order to obtain the first set of location connections, we use derivatives of the basic pattern 'from X to Y'. Using Yahoo! BOSS, we have utilized the API's ability to search for phrases with wildcards. Given a location name $L$ we start the search with patterns "from * to L", "from * * to L". These are Yahoo! BOSS queries where enclosing words in "" means searching for an exact phrase and '*' means a wildcard for exactly one arbitrary word.

This pattern serves a few goals beyond the discovery of connectedness. Thus putting '*'s inside the pattern rather than using "from L to" allowed us to avoid arbitrary truncation of multiword expressions as in 'from Moscow to St. Petersburg' and reduced the probability of capturing unrelated sentence parts like 'from Moscow to cover deficit'.

Location names are usually ambiguous and this type of web queries can lead to a significant amount of noise or location mix. There are two types of ambiguity. First, as in 'from Billericay to Stock....', stock can be a location or a verb. We filter most such cases by allowing only capitalized location names. Besides, such an ambiguity is rarely captured by 'from * to L' patterns. The second type is location ambiguity. Thus 'Moscow' refers to at least 5 location names including farms in Africa and Australia and a locality in Ireland. In order to reduce mixing of locations we use the following simple disambiguation technique. Before performing "from...to" queries, we downloaded up to 100 web pages pointed by each possible *pair* from the given seed locations, generating from a location pair $L_1, L_2$ a conjunctive query "$L_1$ * $L_2$". Then we extracted the most informative terms using a simple probabilistic metric:

$$Rank(w) = \frac{P(w|QueryCorpus)}{P(w|GeneralCorpus)},$$

comparing word distribution in the downloaded $QueryCorpus$ to a large general purpose offline $GeneralCorpus$[1]. We thus obtained a set of

[1]We used the DMOZ corpus (Gabrilovich and Markovitch, 2005).

query-specific disambiguating words. Then we added to all queries the same most frequent word (*DW*) out of the ten words with highest ranks. Thus for the seed set {Moscow, St. Petersburg}, an example of a query is <"from * to Moscow" Russia>.

### 3.2 Iterative location retrieval

We retrieved all search engine snippets for each of these initial queries[2]. If we succeeded to get more than 50 snippets, we did not download the complete documents. In case where only a handful of snippets were obtained, the algorithm downloaded up to 25 documents pointed by these snippets in an attempt to get more pattern instances. In the majority of tested cases, snippets provide enough information for our task, and this information was not significantly extended by downloading the whole documents.

Once we retrieve snippets we identify terms appearing in the snippets in wildcard slots. For example, if the query is <"from * to Moscow" Russia> and we encounter a snippet "...from Vladivostok to Moscow...", we add 'Vladivostok' to our set of seeds. We then continue the search in a breadth first search setting, stopping the search on three conditions: (1) runaway detected – the total frequency of newly obtained terms through some term's patterns is greater than the total frequency of previously discovered terms+seeds. In this case we stop exploration through the problematic term and continue exploration through other terms[3]; (2) we reached a predefined maximal depth $D$[4]; (3) no new terms discovered.

At the end of this stage we get the extended set of terms using the set of snippets where these terms co-appear in patterns.

### 3.3 Enhancement of initial pattern set

In order to get more data, we enhance the pattern set both by discovery of new useful secondary patterns and by narrowing existing patterns. After obtaining the new pattern set we repeat the extraction stage described in Section 3.2.

[2]Yahoo! Boss allows downloading up to a 1000 descriptions, up to 50 in each request. Thus for each seed word, we have performed a few dozen search requests.

[3]Note that the 'problematic' term may be the central term in the region we focus upon – if this happen it means that the seeds do not specify the region well.

[4]Depth is a function of the richness of transport links in the domain. For connected domains (Cardiff, Israel) we used 4, for less connected ones (Nepal) we used 10.

**Adding secondary patterns.** As in a number of previous studies, we improve our results discovering additional patterns from the obtained term set. The algorithm selects a subset of up to 50 discovered $(t_1, t_2)$ term pairs appearing in 'from $t_1$ to $t_2$' patterns and performs the set of additional queries of the form <"$t_1$ * $t_2$" DW>.

We then extract from the obtained snippets the patterns of the form 'Prefix $t_1$ Infix $t_2$ Postfix', where Prefix and Postfix should contain either a punctuation symbol or 1-3 words. Prefix/Postfix should also be bounded from left/right by a punctuation or one of the 50 most frequent words in the language (based on word counts in the offline general corpus). Infix should contain 1-3 words with the possible addition of punctuation symbols[5].

We examine patterns and select useful ones according to the following ranking scheme, based on how well each pattern captures named entities. For each discovered pattern we scan the obtained snippets and offline general corpus for instances where this pattern connects one of the original or discovered location terms to some other term. Let $T$ be the set of all one to three word terms in the language, $T_d \subset T$ the set of discovered terms, $T_c \subset T$ the set of all capitalized terms and $Pat(t_1, t_2)$ indicates one or more co-appearances of $t_1$ and $t_2$ in pattern $Pat$ in the retrieved snippets or offline general corpus. The rank $R$ of pattern $Pat$ is defined by:

$$R(Pat) = \frac{|\{Pat|Pat(t_1, t_2), t_1 \in T_c, t_2 \in T_d\}|}{|\{Pat|Pat(t_1, t_2), t_1 \in T, t_2 \in T_d\}|}$$

In other words, we rank patterns according to the percentage of capitalized words connected by this pattern. We sort patterns by rank and select the top 20% patterns. Once we have discovered a new pattern set, we repeat the term extraction in Section 3.2. We do this only once and not reiterate this loop in order to avoid potential runaway problems. Obtained secondary patterns include different from/to templates "to X from Y by bus"; time/distance combinations "X -N km bus- Y", "X (bus, N min) Y" or patterns in different languages with English location/transport names.

**Narrowing existing patterns.** When available data volume is high, we would like to take advantage of more data by utilizing more specific pattern

[5]Search engines do not support punctuation in queries, hence these symbols were omitted in web requests and considered only when processing the retrieved snippets.

sets. Since Yahoo! allows to obtain only the first 1K snippets, in case we get more than 10K hits, we extend our queries by adding the most common term sequences appearing before or after the pattern. Thus if for the query "from * to Moscow" we got more than 10K hits and among the snippets we see '... bus from X to Moscow...' we create an extended pattern 'bus from * to Moscow' and use the term extraction in Section 3.2 to get additional terms. Unlike the extraction of secondary patterns, this narrowing process can be repeated recursively as long as a query brings more than 10K results.

### 3.4 Extraction of labeled connections

At the end of the discovery stage we get an extended set of patterns and a list of search engine snippets discovered using these patterns. Each snippet which captures terms $t_1, t_2$ in either primary 'from $t_1$ to $t_2$' or secondary patterns represents a potential connection between entities.

Using an observed property of the primary pattern, we select as a label a term or set of terms appearing directly before 'from' and delimited with some high frequency word or punctuation. For example, labels for snippets based on 'from...to' patterns and containing 'the road from...', 'got a bus from', 'a TransSiberian train from...' would be road, bus and TransSiberian train.

Once we acquire labels for the primary patterns, we also attempt to find labels in snippets obtained for secondary patterns discovered as described in Section 3.3. We first locate some already labeled pairs in secondary patterns' snippets where we can see both label and the labeled term pair. Then, based on the label's position in this snippet, we define a label slot position for this type of snippet. Suppose that during the labeling of primary pattern snippets we assigned the label 'bus' to the pair (Novgorod, Moscow) and during the pattern extension stage the algorithm discovered a pattern $P_{new}$ = 'ride to $t_2$ from $t_1$,' with a corresponding snippet '... getting bus ride to Moscow from Novgorod...'. Then using the labeled pair our algorithm defines the label slot in such a snippet type: 'getting [label] ride to $t_2$ from $t_1$'. Once a label slot is defined, all other pairs captured by $P_{new}$ can be successfully labeled.

### 3.5 Merging connection labels

Some labels may denote the same type of connection. Also, large sets of connections can share the same set of transport types. In this case it is desired to assign a single label for a shared set of transports. We do this by a simple merging technique. Let $C_1, C_2$ be sets of pairs assigned to labels $L_1, L_2$. We merge two labels if one of the following conditions holds:

$$(1)|C_1 \cap C_2| > 0.75 * min(|C_1|, |C_2|)$$

$$(2)|C_1 \cap C_2| > 0.45 * max(|C_1|, |C_2|)$$

Thus, either one group is nearly included in the other or each group shares nearly half with the other group. We apply this rule only once and do not iterate recursively. At this stage we also dismiss weakly populated labels, keeping the 10 most populated labels.

### 3.6 Processing of connection graph

Now once we have merged and assigned the labels we create a pattern graph for each label and attempt to clean the graph of noise and unnecessary edges. Our graph definition follows (Widdows and Dorow, 2002). In our pattern graph for label $L$, nodes represent terms and directed edges represent co-appearance of two terms in some pattern in snippet labeled by $L$. We do not add unlabeled snippets to the graph. Now we use a set of techniques to reduce noise and unnecessary edges.

#### 3.6.1 Transitivity elimination

One of the main problems with the pattern-based graph is transitivity of connections. Thus if location A is connected to B and B to C, we frequently acquire a "shortcut" edge connecting A to C. Such an edge diminishes our ability to create a clear and meaningful spatial graph. In order to reduce such edges we employ the following two strategies.

First, neighboring places frequently form fully connected subgraphs. We would like to simplify such cliques to reduce the amount of transitive connections. If three overlapping sets of nodes $\{A_1 \dots A_{n-2}\}$,$\{A_2 \dots A_{n-1}\}$,$\{A_3 \dots A_n\}$ form three different cliques, then we remove all edges between $A_1$ and the nodes in the third clique and between $A_n$ and the nodes in the first clique.

Second, in paths obtained by directional patterns, it is common that if there is a path $A_1 \rightarrow A_2 \cdots \rightarrow A_n$ where $A_1$ and $A_n$ are some major 'key' locations[6], then each of the nodes $A_2 \dots A_{n-1}$ tend to be connected both to $A_1$ and

[6]Such locations will be shown in double circles in the evaluation.

to $A_n$ while intermediate nodes are usually connected only to their close neighbors. We would like to eliminate such transitive edges leaving only the inter-neighbor connections.

We define as key nodes in a graph, nodes whose degree is more than 1.5 times the average graph degree. Then we eliminate the transitive connections: if $A_1$ is a key node and $A_1$ is connected to each of the nodes $A_2 \dots A_{n-1}$, and $\forall i \in \{2 \dots n-1\}, A_i$ is connected to $A_{i+1}$, then we remove the connection of $A_1$ to all nodes $A_3 \dots A_{n-1}$ , leaving $A_1$ only connected to $A_2$.

#### 3.6.2 Clearing noise and merging names

Finally we remove potential noise which accidentally connects remote graph parts. If some edge discovered through a single pattern instance connects distant (distance>3) parts of the graph we remove it.

Additionally, we would like to merge common name alternations and misspellings of places. We merge two nodes A and B into one node if either (1) A, B have exactly the same edges, and their edge count is greater than 2; or (2) edges of A are subset of B's edges and the string edit distance between A and B is less than a third of min(StringLength(A), StringLength(B)).

## 4 Evaluation

Since our problem definition differs significantly from available related work, it is not possible to make direct comparisons. We selected three different cases (in Nepal, Israel, Wales) where the obtained information can be reliably verified, and applied our framework on these settings. As a development set, we used the Russian rail network.

We have estimated the quality of our framework using several measures and observations. First, we calculated the precision and quantity of obtained locations using map information. Then we manually estimated precision of the proposed edges and their labels, comparing them with factual information obtained from maps[7], transport companies and tourist sites. Finally we visually compared a natural drawing of the obtained graph with a real map. In addition, while our goals differ, the third evaluation setting has deliberate significant similarities to (Schockaert et al., 2008), which allows us to make some comparisons.

[7] We recognize that in case of some labels, e.g. 'walk', the precision measure is subjective. Nevertheless it provides a good indication for the quality of our results.

### 4.1 The Annapurna trek area

One of the most famous sites in Nepal is the Annapurna trekking circuit. This is a 14-21 day walking path which passes many villages. Most of the tourists going through this path spend weeks in prior information mining and preparations. However, even when using the most recent maps and guides, they discover that available geographical knowledge is far from being complete and precise. This trek is a good example of a case when formal information is lacking while free-text shared experience in the web is abundant. Our goal was to test whether the algorithm can discover such knowledge automatically starting from few seed location names (we used Pokhara, which is one of the central cities in the area, and Khudi, a small village). The quality of results for this task was very good. While even crude recall estimation is very hard for this type of task, we have discovered 100% of the Annapurna trek settlements with population over 1K, all of the flight and bus connections, and about 80% of the walking connections.

On Figure 1 we can compare the real map and the obtained map[8]. This discovered map includes a partial map[9] for 4 labels – flights, trek, bus and jeep. You can see on the map different lines for each label. The algorithm discovered 132 entities, all of them Annapurna-related locations. This includes correctly recognized typos and alternative spellings, and the average was 1.2 names per place. For example for Besisahar and Pokhara the following spellings were recognized based both on string distance and spatial collocation: *Besishahar, Bensisahar, BesiSahar, Besi Sahar, Beshishahar, Beisahar, Phokra, Pohkala, Pokahara, Pokhara, Pokhar, Pokra, Pokhura, Pokhra*.

We estimated correctness of edges comparing to existing detailed maps. 95% of the edges were correctly placed and labeled. Results were good since this site is well covered and also not very interconnected – most of it is connected in a single possible way. After the elimination process described in the previous section, only 6% of the nodes participate in 3-cliques. Thus, due to the linearity of the original path, our method success-

[8] Graph nodes were manually positioned such that edges do not intersect. Recall that our goal is to build a network graph, which is an abstract structure. The 2-D embedding of the graph shown here is only for illustrative purposes and is not part of our algorithm.

[9] A few dozens of correctly discovered places were omitted to make the picture readable.

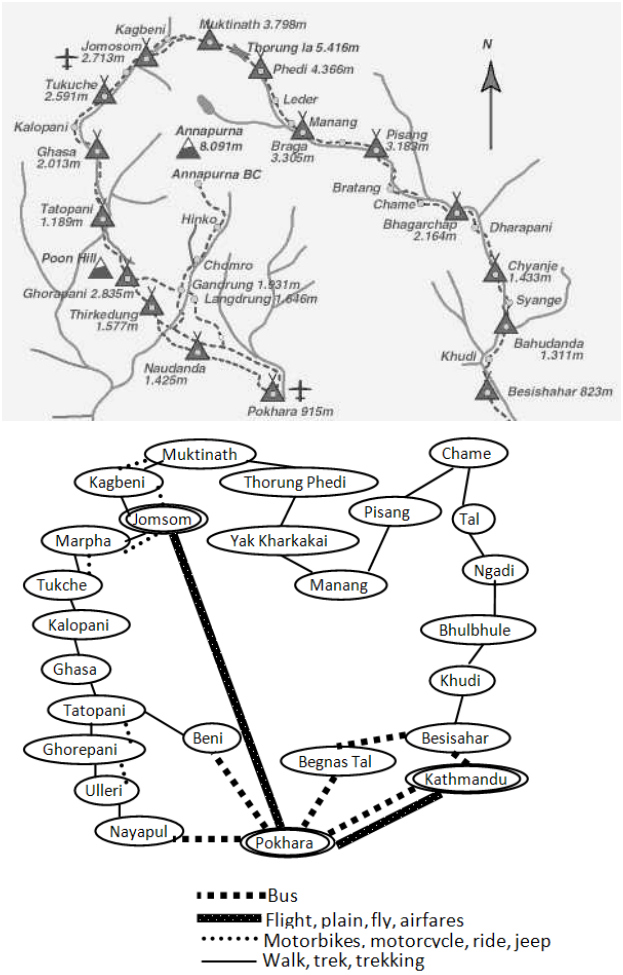


Figure 1: Real path map of Annapurna circuit (above) compared to automatically acquired graph (below). The graph nodes were manually positioned such that edges do not cross each other. Dozens of correctly discovered places were omitted for readability. Double circles indicate key nodes as explained in section 3.6.1

fully avoided the problem of mixing transitively connected nodes into one large clique.

### 4.2 The Israeli south

The southern part of Israel (mostly the Negev desert) is a sparsely populated region containing a few main roads and a few dozen towns. There is a limited number of tourists sites in the Negev and hence little web information is supposed to be available. Our goal was to see if the algorithm can successfully detect at least major entities and to discover their connectedness.

We discovered 56 names of different places, of them 50 correctly belong to the region, where the region is defined as south from the Ashquelon-Jerusalem-Yericho line, the other 6 were Israeli cities/locations outside the region (Tiberias, Metulla, Ben Gurion, Tel Aviv, Ashdod, Haifa). In addition we discovered 23 alternative names for some of the 56 places. We also constructed the corresponding connectedness graphs.

We tested the usefulness of this data attempting to find the discovered terms in the NGA GEOnet Names Server[10] which is considered one of the most exhaustive geographical resources. We could find in the database only 60% of the correctly discovered English terms denoting towns, so 40% of the terms were discovered by us and ignored by this huge coverage database. We also tested the quality of edges, and found that 80% of the discovered edges were correctly placed and labeled. Figure 2 shows a partial graph of the places obtained for the 'road' label.

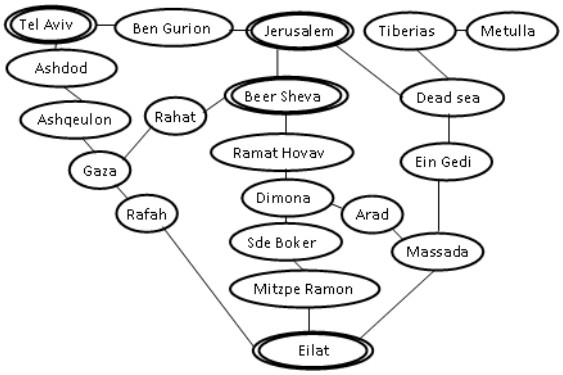


Figure 2: Partial graph for Israel south settings.

### 4.3 The Cardiff area

Cardiff is the capital, largest city and most populous county in Wales. Our goal was to see if we can discover basic means of transport and corresponding locations connected to and inside Cardiff. This exploration also allowed us to compare some of our results to related studies. We executed our algorithm using as seeds Grangetown, Cardiff and Barry. Table 1 shows the most utilized merged labels obtained for most edge-populated graphs together with graph size and estimated precision. In case of flights, treks and trains, precision was estimated using exact data. In other cases we estimated precision based on reading relevant web pages. We can see that the majority of connectivity sets are meaningful and the precision obtained for most of these sets is high. Figure 3 shows a partial graph for 'walking'-type labels and Figure

[10] http://earth-info.nga.mil/gns/html/

| Nodes | Edges(Prec) | Label |
|---|---|---|
| 88 | 120(81) | walking,walk,cycling,short ride taxis, Short bus ride,short walk |
| 131 | 140(95) | flights, airlines,# flights a day |
| 12 | 16(100) | foot path, trek, walking # miles |
| 36 | 51(89) | train, railway, rail travel,rail |
| 32 | 98(65) | bus, road, drive,direct bus |

Table 1: The merged labels obtained for 5 most edge-populated graphs, including number of nodes and edges for each label. The estimated precision according to each label definition is shown in parentheses.

4 shows such a graph for train labels[11]. Comparing the obtained map with real map data we notice a definite correlation between actual and induced relative connection of discovered places.

(Schockaert et al., 2008) used their framework to discover neighborhoods of Cardiff. In our case, the most appropriate relation which connects neighborhood locations is walking/cycling. Hence, comparing the results to previous work, we have examined the results obtained for the 'walking' label in details. (Schockaert et al., 2008) report discovery of 68 locations, of them 7 are alternate entries, 4 can be considered vernacular or colloquial, 10 are not considered to be neighborhoods, and 5 are either close to, but not within, Cardiff, or are areas within Cardiff that are not recognized neighborhoods. In our set we have discovered 88 neighborhood names, of them 18 are alternate entries of correct neighborhoods, 4 can be considered vernacular or colloquial, 3 are not considered to be neighborhoods, and 15 are areas outside the Cardiff area.

Considering alternate entries as hits, we got superior precision of $66/88 = 0.75$ in comparison to $49/68 = 0.72$. It should be noted however that we found many more alternative names possibly due to our larger coverage. Also both our framework and the goal were substantially different.

## 5 Discussion

In this paper we presented a framework which, given a small set of seed terms describing a geographical region, discovers an underlying connectivity and transport graph together with the extraction of common and alternative location names in this region. Our framework is based on the observation that 'from...to'-like patterns can encode connectedness in very precise manner. In our framework, we have combined iterative pattern- and web-based relationship acquisition with the discovery of new patterns and refinement of the location graph. In our evaluation we showed that our framework is capable of extracting high quality non-trivial information from free text given very restricted input and not relying on any heavy preprocessing techniques such as parsing or NER.

The success of the proposed framework opens many challenging directions for its enhancement. Thus we would like to incorporate in our network patterns which allow traveling times and distances to be extracted, such as 'N miles from X to Y'. While in this paper we focus on specific type of geographical relationships, similar frameworks can be useful for a wider class of spatial relationships. Automated acquisition of spatial data can significantly help many NLP tasks, e.g., question answering. We would also like to incorporate some patterns based on (Egenhofer and Shariff, 1998), such as 'crosses', 'goes through' or 'runs into', which may allow automated acquisition of complex spatial relationships. Finally, we would like to incorporate in our framework mod-

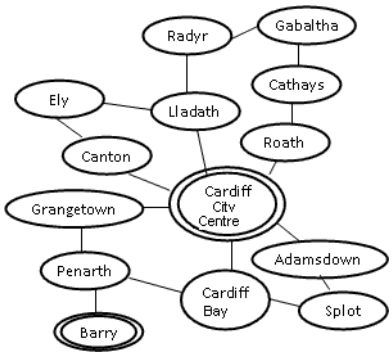


Figure 3: Partial graph of the obtained Cardiff region for the walk/walking/cycling label.

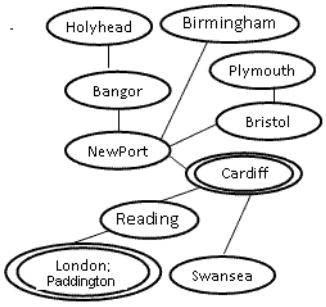


Figure 4: Partial graph of the obtained Cardiff region for the railway/train label.

[11] Spatial position of displayed graph components is arbitrary, we only made sure that there are no intersecting edges.

ules which may allow recognition of structured data, like those developed by (Schockaert et al., 2008).

## References


Eugene Agichtein, Luis Gravano, 2000. Snowball: Extracting Relations from Large Plain-text Collections. *ACM DL '00*.

Karla Borges, Alberto Laender, Claudia Medeiros, Clodoveu Davis, 2007. Discovering Geographic Locations in Web Pages Using Urban Addresses. *Fourth Workshop on Geographic Information Retrieval*.

Sharon Caraballo, 1999. Automatic Construction of a Hypernym-labeled Noun Hierarchy from Text. *ACL '99*.

Timothy Chklovski, Patrick Pantel 2004. VerbOcean: Mining the Web for Fine-grained Semantic Verb Relations. *EMNLP '04*.

James R. Curran, Marc Moens, 2002. Improvements in Automatic Thesaurus Extraction. *SIGLEX '02*.

Dmitry Davidov, Ari Rappoport, 2006. Efficient Unsupervised Discovery of Word Categories Using Symmetric Patterns and High Frequency Words. *COLING-ACL '06.*

Dmitry Davidov, Ari Rappoport, Moshe Koppel, 2007. Fully Unsupervised Discovery of Concept-specific Relationships by Web Mining. *ACL '07*.

Dmitry Davidov, Ari Rappoport, 2008. Classification of Semantic Relationships Between Nominals Using Pattern Clusters. *ACL '08.*

Max Egenhofer, Rashid Shariff, 1995. Naive Geography. *Proceedings of COSIT '95.*

Max Egenhofer, Rashid Shariff, 1998. Metric Details for Natural-Language Spatial Relations. *Journal of the ACM TOIS*, 4:295–321, 1998.

Oren Etzioni, Michael Cafarella, Doug Downey, Stanley Kok, Ana-maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, Alexander Yates, 2004. Web-scale Information Extraction in KnowItAll. *WWW '04*.

Evgeniy Gabrilovich, Shaul Markovitch, 2005. Feature Generation for Text Categorization Using World Knowledge. *IJCAI '05*.

James Gorman, James R. Curran, 2006. Scaling Distributional Similarity to Large Corpora. *COLING-ACL '06*.

Marti Hearst, 1992. Automatic Acquisition of Hyponyms from Large Text Corpora. *COLING '92*.

Dekang Lin, 1998. Automatic Retrieval and Clustering of Similar Words. *COLING '98*.

Simon Overell, Stefan Ruger, 2007. Geographic Co-occurrence as a Tool for GIR. *Fourth ACM Workshop on Geographical information retrieval*.

Patrick Pantel, Dekang Lin, 2002. Discovering Word Senses from Text. *SIGKDD '02*.

Patrick Pantel, Deepak Ravichandran, Eduard Hovy, 2004. Towards Terascale Knowledge Acquisition. *COLING '04*.

Ross Purves, Paul Clough, Christopher Jones, Avi Arampatzis, Benedicte Bucher, Gaihua Fu, Hideo Joho, Awase Syed, Subodh Vaid, Bisheng Yang, 2007. The Design and Implementation of SPIRIT: a Spatially Aware Search Engine for Information Retrieval on the Internet. *International Journal of Geographical Information Science*, 21(7):717-745.

Ellen Riloff, Rosie Jones, 1999. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. *AAAI '99*.

Diana Santos, Nuno Cardoso, 2008. GikiP: Evaluating Geographical Answers from Wikipedia. *Fifth Workshop on Geographic Information Retrieval*.

Steven Schockaert, Philip Smart, Alia Abdelmoty, Christopher Jone, 2008. Mining Topological Relations from the Web. *International Workshop on Flexible Database and Information System Technology*, workshop at DEXA, Turin, pp. 652–656.

Dominic Widdows, Beate Dorow, 2002. A Graph Model for Unsupervised Lexical Acquisition. *COLING '02*.