Relation Extraction from Text

Paper Implementation - Mentor: Dr. Qi Li

Avinash Kadimisetty
University of Illinois at Urbana-Champaign
avinash9@illinois.edu

Ankit Kumar
University of Illinois at Urbana-Champaign
ankitk3@illinois.edu

ABSTRACT

As the amount of text data in various forms like news articles, medical records, social media messages and posts is increasing everyday there is an apparent need to convert into a structured data for uses in various applications like Search, Question Answering etc. Relation extraction is a potential problem solver where relations between different entities can be generated. For example, in the text Barack Obama and Michelle Obama have celebrated their 27th marriage anniversary, Michelle Obama can be related to Barack Obama with wife of relationship. The goal of Relation Extraction is to generate many of such relations from a given text and store these relations in a structured way. Domain Knowledge can also be used to generate more meaningful relations. For example, to understand semantic relations between disease and treatment from biomedical text. The objective of this project is to do a comprehensive review of the existing distance based and pattern based relation extraction techniques, compare their strengths and weaknesses, and evaluate their performance. In this project we are implementing an approach towards extracting relations from test - PATTY: A Taxonomy of Relational Patterns with Semantic Types [11].

ACM Reference Format:
Avinash Kadimisetty and Ankit Kumar. 2018. Relation Extraction from Text: Paper Implementation - Mentor: Dr. Qi Li. In *Proceedings of* . ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The PATTY system [11], which is a huge resource for textual patterns denoting binary relations between entities is based on efficient frequent itemset mining algorithms that can process web-scale corpora. The taxonomy comprises of 350,569 pattern synsets. PATTY aims to systematically compile relational patterns and impose a semantically typed structure on them similar to Wordnet-style taxonomy of binary relations. In wordnet, all the nouns, verbs, adjectives are grouped into sets of synonyms and these are then arranged based on the taxonomy of hypernyms. Most large scale knowledge bases are mostly limited to extracting binary relationships between entities but these do not consider textual patterns. In PATTY, patterns that contain semantic types such as ⟨*singer*⟩ *sings* ⟨*song*⟩ are created. All the syntactic variations such as ⟨*singer*⟩ *sings her* ⟨*song*⟩ and ⟨*singer*⟩ *sings* ⟨*song*⟩ are automatically generalized into a more general pattern ⟨*sings*⟩ *[Parts of Speech]* ⟨*song*⟩. Such organization could be more challenging because the number of possible patterns increases exponentially with the length of the patterns. Also, if the corpus is small, the different patterns may apply to the same set of entity pairs in the corpus. In addition, computing the mutual subsumptions on a large set of patterns may be slow.

The novelties of PATTY are as follows - An expressive family of relational patterns which combines syntactic features, ontological type signatures and lexical features (SOL) are defined. An efficient scalable algorithms than infer SOL patterns and subsumptions at scale is presented and is based on instance-level overlaps and an ontological type hierarchy. The Wikipedia corpus is used in PATTY and a total of 350,569 pattern synsets are obtained with 84.7% precision.

1.1 Flow

The next few sections describe the PATTY algorithm with respect to YAGO2 knowledge base and the sections later describe the implementation specifics with respect to the Comparative Toxicogenomics Database.

2 PATTY ALGORITHM

The PATTY algorithm has four phases.

(1) **Pattern Extraction:** A pattern is a surface string that occurs between a pair of entities in a sentence, thus the first step is to obtain basic textual patterns from the input corpus.
(2) **SOL Pattern Transformation:**The second step is to transform plain patterns into SOL patterns thereby enhancing them with ontological types.
(3) **Pattern Generalization:** The third step is to generalize the patterns, both syntactically and semantically.
(4) **Subsumption Mining:** The last step is to arrange the patterns into a hierarchy based on hypernymy/hyponymy relations between patterns.

A snapshot depicting the entire flow of the PATTY algorithm is given shown in the figure 1. An example type system is shown in figure 2

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

, ,
© 2018 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM. . . $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

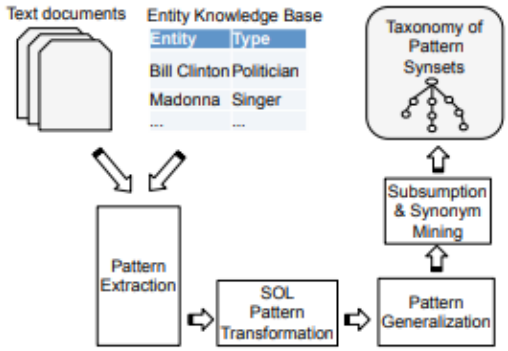

Figure 1: PATTY flow

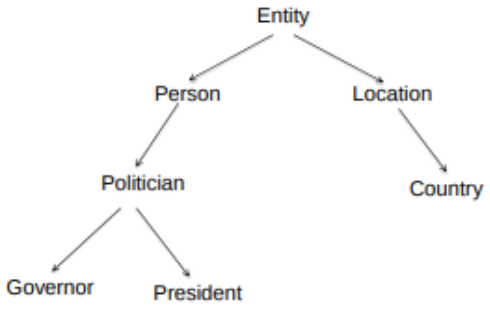

Figure 2: An example Type System

2.1 Pattern Extraction Algorithm

Given an input corpus, the Stanford parser [4] is applied to the individual sentences of the corpus to obtain the dependency paths which form a directed graph, with words being the nodes and dependencies being the edges. For example, the sentence, *Winehouse effortlessly performed her song Rehab.* yields the following dependency paths. The algorithm is shown in Algorithm 1

nsubj(performed-3, Winehouse-1)
advmod(performed-3, effortlessly-2)
poss(Rehab-6, her-4)
nn(Rehab-6, song-5)
dobj(performed-3, Rehab-6)

By using a dictionary of entities, named entities are detected. Using the YAGO2 [7] knowledge base, the noun phrases containing at least one proper noun are matched against the dictionary. For example, for the above sentence, entity detection yields Amy Winehouse and Rehab (song). Whenever two named entities appear in the same sentence, a

Algorithm 1 Extract Patterns

```
1:  procedure EXTRACTPATTERNS
2:      C ← All the sentences from the given corpus
3:      for c ∈ C do
4:          Entities ← Detect named entities in c
5:          if ‖Entities‖ ≥ 2 then
6:              G ← GenerateDependencyGraph(c)
7:              P ← GetDependencyPaths(∀e_i, e_j ∈ Entities)
8:              for p ∈ P do
9:                  emit(e_i, p, e_j, pos)
10:             end for
11:         end if
12:     end for
13: end procedure
```

textual pattern is extracted. Then, by traversing the dependency graph the shortest path connecting the two entities is found out. In the above example, the shortest path between the entities Winehouse and Rehab is Winehosue nsubj performed dobj Rehab. Since, PATTY aims to capture only relations that refer to the subject-relation-object triples, only the shortest paths that start with subject-like dependencies such as nsubj, rcmod and partmod are considered. To reflect the full meaning of patterns, the shortest path is expanded using the adverbial and adjectival modifiers. The final textual pattern comprises of the words from the expanded shortest path. In the above example, the textual pattern is Amy Winehouse effortlessly performed Rehab (song).

2.2 SOL Pattern Generation

The textual patterns are transformed into a new type of patterns called the SOL patterns. SOL patterns extend lexico-syntactic patterns by ontological type signatures for entities. Typically, the SOL pattern is a sequence of words, Parts of Speech (POS) tags, wildcards and ontological types. A special POS tag [word] which denotes any word of any POS class is used in the patterns. For example, a tag [verb] to denote any word which is a verb. The wildcard, denoted as * stands for any sequence of words. These are essentially to avoid overfitting of patterns to the corpus. An ontological type is a semantic class name (such as ⟨singer⟩) that stands for an instance of that class. At least two types are present in every patterns and they are designated as entity placeholders. To explain these terms, consider a pattern ⟨person⟩s [adj] voice * ⟨song⟩. Here ⟨person⟩ and ⟨song⟩ are the entity placeholders. The sentence Amy Winehouses sweet voice in Rehab and Elvis Presleys soft voice in song All shook up are examples of strings in the corpus that match the pattern. The support set for these two patterns is (Amy, Rehab), (Elvis, AllShookUp), where each pair is called the support pair.

Pattern B is *syntactically more general* than pattern A, if every string matching A also matches pattern B. Pattern B is *semantically more general* than Pattern A if the support set of B is a superset of the support set of Pattern A. If A and B are both semantically more general to each other,

then the patterns are synonymous and the set of synonymous patterns is called a pattern *synset*. Two patterns are called *semantically different* if neither of them is semantically more general than other.

The SOL patterns from the textual patterns are generated by decomposing the textual patterns into n-grams. A SOL pattern contains only n-grams that appear frequently in the corpus and the remaining sequence of words are replaced by the wildcards (*). The frequent n-grams are efficiently found out using the Apriori technique where each sentence is viewed as a transaction with a purchase of several n-grams. The algorithm for finding n-grams is shown in Algorithm 2. These n-grams allow a sentence to be broken down into wildcard separated subsequences which gives an SOL pattern.

Algorithm 2 Mining n-grams

```
1: procedure MINENGRAMS
2:     P ← List of all textual patterns
3:     S ← Generate a sequential database of sequences.
4:     N = dictionary()
5:     for each sequence Seq in S do
6:         for each 3 length n-gram K in Seq do
7:             N[K] += 1
8:         end for
9:     end for
10:    N-grams = []
11:    for n-gram K in N.keys() do
12:        if N[K] ≥ 5 then
13:            add K to N-grams
14:        end if
15:    end for
16:    return N-grams
17: end procedure
```

The statistical strength of a pattern is quantified by means of its support set. The support of a pattern p which has a type signature t1xt2 is the size of its support set. For confidence, the support-set sizes of p and an untyped variant p^u of p, in which the types ⟨t1⟩ and ⟨t2⟩ are replaced by the generic type ⟨entity⟩. The confidence is then defined as the ratio of support-set sizes of p and p^u.

2.3 Syntactic Pattern Generalization

Most of the patterns can be generalized into a syntactically more general pattern by using POS tags to replace words, by combining more n-grams and replacing them with wild cards (*) or by using generic entity types in the textual pattern. The quality of generalizations is difficult to assess as a generalization could subsume two semantically different patterns. For example ⟨person⟩ [vb] ⟨person⟩ subsumes two semantically different patterns ⟨person⟩ loves ⟨person⟩ and ⟨person⟩ hates ⟨person⟩. Such a generalization will lead to misrepresenting the textual pattern and is meaningless. Thus, for every pattern, all possible generalizations are generated. If a generalization subsumes multiple patterns with disjoint support sets, we discard the generalized pattern.

2.4 Taxonomy Construction Algorithm

After extracting the patterns from the corpus, they have to be arranged in a semantic taxonomy. An expensive solution is to compare every patterns support-set with every other pattern support-set to determine the inclusion, independence or mutual inclusion. Since this process is very slow, a prefix tree (used in FP Growth algorithm) can be used. The support sets of the patterns are stored in the prefix-tree. An algorithm is then developed to obtain set intersections from the prefix tree.

Lets consider the table of pattern synsets and their support sets as shown in Table 1. An entity pair in a support set is denoted by a letter. For example, the entry A,80 in the support set of the pattern ⟨Politician⟩ was governor of ⟨State⟩ may denote the pair *Arnold Schwarzenegger, California* with a frequency of 80.

ID	Pattern Synset & Support Sets
P_1	⟨Politician⟩ was governor of ⟨State⟩ A,80 B,75 C,70
P_2	⟨Politician⟩ politician from ⟨State⟩ A,80 B,75 C,70 D,66 E, 64
P_3	⟨Person⟩ daughter of ⟨Person⟩ F,78 G,75 H,66
P_4	⟨Person⟩ child of ⟨Person⟩ I,88 J,87 F,78 G,75 K,64

Table 1: Pattern synsets and their support sets

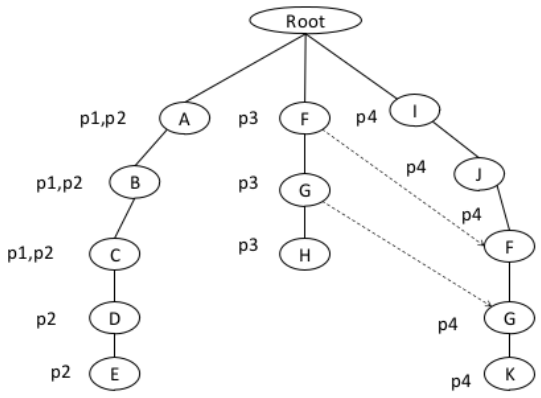

Figure 3: Prefix Tree

The contents of the support sets are used to construct the prefix-tree, where each node is an entity-pair. If two synsets have entity pairs in common then they share a common prefix and thus the shared parts are represented by one prefix-path in the prefix-tree. Thus, by representing the tree in a compact manner, the subsumptions can be directly read from the tree. The entity pairs are inserted into the tree in the decreasing

order of their frequency to increase the chance of shared prefixes.

Once the prefix-tree is constructed, the subsumptions can be efficiently mined by avoiding the comparison of every path to other parth as this introduces the same inefficiencies that the traditional approach suffers from. It follows from the tree construction that, for any node N in the tree, all paths containing N can be found by following node N's links including the same entity-pair links. We can reach all the pattern synsets which share common nodes with a synset P by traversing the entire path of the synset P. We can determine which paths are subsumed by a synset P if we start traversing the tree in bottom up fashion, starting at the last node in the support set of P.

Once the subsumptions between relational patterns are generated, we need to remove the cycles in the graph generated. In other words, a minimal total number of subsumptions whose removal results in a directed acyclic graph. This is a NP hard problem but we use a greedy algorithm for removing cycles and eliminating redundancy in the subsumptions, thus effectively constructing a directed acyclic graph. The DAG of pattern synsets is the PATTY taxonomy.

3 IMPLEMENTATION

In this section, we discuss how we adapted PATTY framework for our dataset. We implemented it on CTD dataset [3]. The GitHub code is available at (https://github.com/ankitk28/CS412Project)

3.1 Data setup

The given corpus is related to Chemical, Genes and Diseases which has 302,735 sentences. The entities are already typed and matched in the given corpus and we considered only the entities Chemical, Gene and Disease to generate the textual patterns. Using *spacy* [2] library as our *NLP* utility, we implemented different phases of the algorithm. The given corpus is first read into the environment and each sentence is checked for the presence of entities. If a sentence doesn't have a minimum of two entities it is discarded and is not used for generating textual patterns. Since the entities are already typed in the sentence, the detection is done by checking for the presence of any of the words Chemical or Gene or Disease. Lets call the set of sentences which have a minimum of two entities as S.

3.2 Extracting textual patterns

To generate textual patterns for each sentence in S, we need to find out the shortest dependecy path between two entities. If a sentence has more than two entities, every possible pair of entities is considered as a combination. For each sentence, the shortest path is found by using *spacy's* dependency parser as the first step. Using the parsed sentence, we constrcuted a network graph by children of each token in the sentence and the token itself as edges. Using the *networkx* module in python, the shortest path between two entities is found out. The dependency paths are further extended by using *advmod* dependencies. This process, when run on each and every sentences in S generates the textual patterns T.

3.3 Generating SOL patterns

The next part is generating SOL patterns which has entity types and wildcards. To generate the frequent n-grams, we created a sequential database in which each sequence is a word sequences that occurs between two entities. Since the paper uses only n-grams of length 3, we generated n-grams of only length 3. To generate the frequent n-grams we used a sliding window based sequence generator. For each sequence in the database, we used a sliding window of size three to observe all the sequences of length 3 and stored the frequency of each sequence in a dictionary. The paper didn't specify a minimum support threshold and we assumed the threshold to be 5 occurrences. Using the dictionary of length 3 n-grams created above, we generated the frequent n-grams by extracting the n-grams which satisfy the minimum support threshold.

Now, we have to replace all the word sequences in each pattern which are not n-grams as wildcards (*). For each sequence, we split the sentence by the presence of n-grams and entities and remove the n-grams from the list of textual patterns. This way we were able to identity the non n-grams from each textual pattern. Then, each non n-gram from every textual pattern is replaced by *. To find the entity type, in each textual pattern we checked those words which starts with either of Chemical or Gene or Disease and then replaced those words with an entity type to get a typed pattern. This way we were able to generate all the typed patterns.

3.4 Generalizing the SOL patterns

For a given pattern P, we did 3 syntactic generalizations:

- Contraction of n- gram: Generalized pattern P' is obtained from P by contracting 1 n-gram in P
- POS replacement of a word: Generalized pattern P' is obtained from P by 1 word by its part of speech tag in P (except entity)
- POS replacement of all words: Generalized pattern P' is obtained from P by replacing all words (except entity) by its part of speech tag in P
- Generalizing type: Generalized pattern P' is obtained from P by generalizing type signature in the pattern P (example: ⟨Chemical⟩ is replaced by ⟨Entity⟩)

We further reject those syntactic generalization which subsumes 2 patterns with disjoint subsets (because that would imply that the generalization is semantically meaningless)

3.5 Pattern strength and confidence

For all the patterns, we first obtain the strength of that pattern which will be the size of the support of that function. Then for all typed pattern we obtain its confidence by normalizing the strength of the typed pattern by the strength of its corresponding untyped pattern.

Pattern	Confidence	Statistical Strength
⟨CHEMICAL⟩ * apoptosis ADP cells ⟨DISEASE⟩	0.52	105
⟨CHEMICAL⟩ * apoptosis [WORD] cells ⟨DISEASE⟩	0.52	105
⟨CHEMICAL⟩ ADP effects on * ⟨DISEASE⟩	0.38	91
⟨CHEMICAL⟩ ADP effects on * ⟨GENE⟩	0.31	73
⟨CHEMICAL⟩ of effects on * ⟨GENE⟩	0.30	73
⟨CHEMICAL⟩ ADP effect on * ⟨GENE⟩	0.47	65
⟨CHEMICAL⟩ of effect on * ⟨GENE⟩	0.44	65
⟨CHEMICAL⟩ ADP effects on * ⟨CHEMICAL⟩	0.24	58
⟨CHEMICAL⟩ of effects on ⟨DISEASE⟩	0.66	48
⟨CHEMICAL⟩ ADP effects on ⟨DISEASE⟩	0.65	48

Table 2: Top 10 frequent patterns based on statistical strength for support threshold 5

Pattern	Confidence	Statistical Strength
⟨CHEMICAL⟩ * ADP effect had ⟨CHEMICAL⟩	1.0	14
⟨CHEMICAL⟩ * [WORD] effect had ⟨CHEMICAL⟩	1.0	14
⟨CHEMICAL⟩ PART exposure after * ⟨GENE⟩	1.0	13
⟨DISEASE⟩ of cause is ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ ADP cause is ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ [WORD] cause is ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ of cause VERB ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ of cause [WORD] ⟨DISEASE⟩	1.0	13
⟨CHEMICAL⟩ induced expression of ⟨GENE⟩	1.0	10
⟨CHEMICAL⟩ induced expression ADP ⟨GENE⟩	1.0	10

Table 3: Top 10 frequent patterns based on confidence for support threshold 5

3.6 Mining subsumptions

We use Algorithm 3 with alpha = 0 and use Wilson score (for 95% confidence interval) for weighting each subsumption. Further we used the optimization using inverst synset data as suggested in [13].

3.7 DAG construction for taxonomy

We sort the subsumption pair by their Wilson scores. Then, as suggested in the original paper, using the greedy algorithm we construct the directed acyclic graph for submsumptions. This process will generate a DAG of pattern synsets which is our final taxonomy.

4 EVALUATION AND DISCUSSION

To evaluate our algorithms we have discussed about the results from several angles. We have implemented the algorithm for various minimum support thresholds for frequent n-gram mining. We have discussed the comparison between the the relations from the knowledge based and the relations generated by the algorithm. In brief, we will discuss about the top frequent patterns based on the statistical strength, top highly confident patterns, patterns at the bottom with least confidence but good statistical strength. We shall also go over some meaningless patterns generated and the subsumptions. During our evaluation we observed that one of the syntactic generalization methods we implemented viz. replacing typed entities with generic entities resulted in discarding such patterns with generic entities. We have given examples to support this statement.

4.1 Setup

The algorithm for extracting and mining relations is run on a text corpus with 300K sentences. The entities are matched and typed using the CTD knowledge base. We used python dictionaries and text files to store the textual patterns and the relations. All phases of Patty algorithm took less than half an hour to run except the phase Pattern Extraction which took around six hours. We now discuss the results we obtained for different minimum support for mining n-gram.

4.2 Results for Threshold 5

The following sections discuss about the quality of the patterns and relations genearted when the minimum support threshold for mining frequent n-grams is set as 5. Using this support value, the PATTY algorithm generated 60,696 patterns and top 10 patterns based on statistical strength and confidence are shown in tables 2 and 3 respectively. The bottom 10 patterns based on statistical strength are shown in table 9 respectively. It can be observed that the number of patterns generated by using low support threshold is far higher than the number of patterns generated using high

Pattern	Confidence	Statistical Strength
⟨CHEMICAL⟩ * apoptosis ADP cells ⟨DISEASE⟩	0.536	109
⟨CHEMICAL⟩ * apoptosis [WORD] cells ⟨DISEASE⟩	0.536	109
⟨CHEMICAL⟩ ADP effects on * ⟨DISEASE⟩	0.387	91
⟨CHEMICAL⟩ ADP effects on * ⟨GENE⟩	0.310	73
⟨CHEMICAL⟩ of effects on * ⟨GENE⟩	0.306	73
⟨CHEMICAL⟩ ADP effect on * ⟨GENE⟩	0.460	65
⟨CHEMICAL⟩ of effect on * ⟨GENE⟩	0.445	65
⟨CHEMICAL⟩ ADP effects on * ⟨CHEMICAL⟩	0.251	59
⟨CHEMICAL⟩ * on effects ADP ⟨CHEMICAL⟩	0.757	53
⟨CHEMICAL⟩ of effects on ⟨DISEASE⟩	0.666	48

Table 4: Top 10 frequent patterns based on statistical strength for support threshold 10

Pattern	Confidence	Statistical Strength
⟨CHEMICAL⟩ * PART NOUN ADP * ⟨GENE⟩	1.0	47
⟨CHEMICAL⟩ * on effect had ⟨CHEMICAL⟩	1.0	15
⟨CHEMICAL⟩ * ADP effect had ⟨CHEMICAL⟩	1.0	15
⟨CHEMICAL⟩ * [WORD] effect had ⟨CHEMICAL⟩	1.0	15
⟨CHEMICAL⟩ * on effect VERB ⟨CHEMICAL⟩	1.0	15
⟨CHEMICAL⟩ PART exposure after * ⟨GENE⟩	1.0	14
⟨DISEASE⟩ of cause is ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ ADP cause is ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ [WORD] cause is ⟨DISEASE⟩	1.0	13
⟨DISEASE⟩ of cause VERB ⟨DISEASE⟩	1.0	13

Table 5: Top 10 frequent patterns based on confidence for support threshold 10

Pattern	Confidence	Statistical Strength
⟨CHEMICAL⟩ of effects on * ⟨GENE⟩	0.307	73
⟨CHEMICAL⟩ of effect on * ⟨GENE⟩	0.445	65
⟨CHEMICAL⟩ of effects on ⟨DISEASE⟩	0.667	48
⟨GENE⟩ expression in cells ⟨DISEASE⟩	0.839	47
⟨CHEMICAL⟩ * growth of cells ⟨DISEASE⟩	0.734	47
⟨CHEMICAL⟩ by inhibition of ⟨GENE⟩	0.811	30
⟨CHEMICAL⟩ of effect in * ⟨DISEASE⟩	0.638	30
⟨CHEMICAL⟩ by induced in * ⟨DISEASE⟩	0.604	29
⟨CHEMICAL⟩ of effect on * ⟨CHEMICAL⟩	0.192	28
⟨CHEMICAL⟩ by inhibition of * ⟨GENE⟩	0.692	27

Table 6: Top 10 frequent patterns before generalization based on statistical strength for support threshold 5

support thresholds. The top 10 frequent patterns have higher statistical strength when the support threshold is lower.

4.3 Results for Threshold 10

The following sections discuss about the quality of the patterns and relations genearted when the minimum support threshold for mining frequent n-grams is set as 10. Using this support value the PATTY algorithm generated 30,917 patterns and the top 10 patterns based on statistical strength and confidence are shown in tables 4 and 5 respectively. The bottom 10 patterns based on statistical strength are shown in table 10 respectively.

4.4 Results for Threshold 20

The following sections discuss about the quality of the patterns and relations generated when the minimum support threshold for mining frequent n-grams is set as 20. Using this support value the PATTY algorithm generated 14881 patterns and the top 10 patterns based on statistical strength and confidence are shown in tables 12 and 13 respectively. The bottom 10 patterns based on statistical strength are shown in table 11 respectively.

Pattern	Confidence	Statistical Strength
⟨*CHEMICAL*⟩ of effects on * ⟨*GENE*⟩	0.307	73
⟨*CHEMICAL*⟩ of effect on * ⟨*GENE*⟩	0.445	65
⟨*CHEMICAL*⟩ of effects on ⟨*DISEASE*⟩	0.667	48
⟨*GENE*⟩ expression in cells ⟨*DISEASE*⟩	0.839	47
⟨*CHEMICAL*⟩ * growth of cells ⟨*DISEASE*⟩	0.734	47
⟨*CHEMICAL*⟩ * in lines cell ⟨*DISEASE*⟩	0.56	42
⟨*CHEMICAL*⟩ by inhibition of ⟨*GENE*⟩	0.811	30
⟨*CHEMICAL*⟩ of effect in * ⟨*DISEASE*⟩	0.638	30
⟨*CHEMICAL*⟩ had effect on * ⟨*GENE*⟩	0.577	30
⟨*CHEMICAL*⟩ by induced in * ⟨*DISEASE*⟩	0.604	29

Table 7: Top 10 frequent patterns before generalization based on statistical strength for support threshold 10

Pattern	Confidence	Statistical Strength
⟨*CHEMICAL*⟩ of effects on * ⟨*GENE*⟩	0.307	73
⟨*CHEMICAL*⟩ of effect on * ⟨*GENE*⟩	0.445	65
⟨*CHEMICAL*⟩ of effects on * ⟨*CHEMICAL*⟩	0.252	60
⟨*CHEMICAL*⟩ * on effects of ⟨*CHEMICAL*⟩	0.764	55
⟨*GENE*⟩ * apoptosis in cells ⟨*DISEASE*⟩	0.28	49
⟨*CHEMICAL*⟩ of effects on ⟨*DISEASE*⟩	0.667	48
⟨*GENE*⟩ expression in cells ⟨*DISEASE*⟩	0.839	47
⟨*CHEMICAL*⟩ * growth of cells ⟨*DISEASE*⟩	0.734	47
⟨*CHEMICAL*⟩ * in lines cell ⟨*DISEASE*⟩	0.56	42
⟨*CHEMICAL*⟩ of effects in * ⟨*DISEASE*⟩	0.631	41

Table 8: Top 10 frequent patterns before generalization based on statistical strength for support threshold 20

Pattern	Confidence	Statistical Strength
⟨*DISEASE*⟩ ⟨*GENE*⟩ * ⟨*DISEASE*⟩	0.001	1
⟨*GENE*⟩ ⟨*DISEASE*⟩ * ⟨*DISEASE*⟩	0.001	1
⟨*DISEASE*⟩ of effects [WORD] * ⟨*DISEASE*⟩	0.002	1
⟨*DISEASE*⟩ of effects ADP * ⟨*DISEASE*⟩	0.003	1
⟨*DISEASE*⟩ of effect [WORD] * ⟨*DISEASE*⟩	0.003	1
⟨*CHEMICAL*⟩ ⟨*GENE*⟩ * [WORD] [WORD] [WORD] ⟨*GENE*⟩	0.003	1
⟨*DISEASE*⟩ ⟨*GENE*⟩ * [WORD] [WORD] [WORD] ⟨*DISEASE*⟩	0.003	1
⟨*GENE*⟩ ⟨*CHEMICAL*⟩ * [WORD] [WORD] [WORD] ⟨*GENE*⟩	0.003	1
⟨*DISEASE*⟩ [WORD] effects on * ⟨*DISEASE*⟩	0.003	1
⟨*DISEASE*⟩ [WORD] [WORD] [WORD] * ⟨*CHEMICAL*⟩ ⟨*DISEASE*⟩	0.004	1

Table 9: Bottom 10 frequent patterns based on statistical strength for support threshold 5

4.5 Discussion

In this section, we will compare the patterns generated by using different support threshold and describe the quality of the patterns and subsumptions generated. From Tables 6, 7 and 8 it can be observed that the top 4 to 5 frequent patterns generated before generalization are the same which is expected but the patterns following them are different across different support thresholds. From the bottom 10 patterns shown in Tables 9, 10 and 11, it can be seen that most of the patterns are meaningless (as we have not done a holistic generalization due to space constraints), but some are indeed useful, for example ⟨*DISEASE*⟩ ADP effects on * ⟨*DISEASE*⟩.

Using syntactic generalization, we have obtained the following as the top 5 subsumptions for support threshold 20 as the following (the second sentence subsumed the first sentence). Most of the subsumptions are sobvious because of syntactic generalization.

- ⟨CHEMICAL⟩ * caused increase in ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * caused increase [WORD] ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * VERB increase in ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * caused increase [WORD] ⟨DISEASE⟩

Pattern	Confidence	Statistical Strength
⟨*DISEASE*⟩ ⟨*GENE*⟩ * ⟨*DISEASE*⟩	0.002	1
⟨*DISEASE*⟩ of effects [WORD] * ⟨*DISEASE*⟩	0.002	1
⟨*DISEASE*⟩ * ⟨*GENE*⟩ * ⟨*DISEASE*⟩	0.002	1
⟨*DISEASE*⟩ of effects ADP * ⟨*DISEASE*⟩	0.003	1
⟨*DISEASE*⟩ of effect [WORD] * ⟨*DISEASE*⟩	0.003	1
⟨*DISEASE*⟩ [WORD] effects on * ⟨*DISEASE*⟩	0.003	1
⟨*GENE*⟩ * NOUN of cells ⟨*GENE*⟩	0.004	1
⟨*GENE*⟩ * [WORD] of cells ⟨*GENE*⟩	0.004	1
⟨*DISEASE*⟩ of effects on * ⟨*DISEASE*⟩	0.004	1
⟨*DISEASE*⟩ ADP effects on * ⟨*DISEASE*⟩	0.004	1

Table 10: Bottom 10 frequent patterns based on statistical strength for support 10

Algorithm 3 Mining Subsumptions

procedure MINESUBSUMPTIONS
Input: Support set prefix-tree T and a subsumption threshold
Output: Complete set of subsumption relations, S
  S ← Φ
  W ← Φ
  for node $n_i \in T$ **do**
    if not n_i is synset terminating **then**
      continue;
    end if
    $C_i \leftarrow$ synsets terminating in n_i
    $S_i \leftarrow \Phi$
    for node $n_j \in$ path $n_i \rightarrow root$ **do**
      $CX_j \leftarrow$ synsets in n_j
      for synset $c_i \in C_i$ **do**
        for synset $cx_j \in CX_j$ **do**
          if $cx_j \implies c_i \notin S_i$ **then**
            $S_i.add(cx_j \implies c_i, 0)$
          end if
          $S_i.increment(cx_j \cap c_i)$
        end for
      end for
      $S \cup \{cx \implies c \in S_i : (|cx - c| \leq \alpha)\}$
      $W \cup \{WilsonScoreof(cx, c, 0.95)\}$
    end for
  end for
  return S
end procedure

- ⟨CHEMICAL⟩ * caused increase ADP ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * caused increase [WORD] ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * caused increase in ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * caused increase ADP ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * VERB increase in ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * caused increase ADP ⟨DISEASE⟩

Disregarding syntactic generalization, we get the following subsumptions which are very meaningful. For example, "exposure associated to" relation subsumes "of effects to" for a particular textual pattern. The top 5 subsumptions without using syntactic generalization are shown below.

- ⟨CHEMICAL⟩ to exposure associated ⟨CHEMICAL⟩
  ⟨CHEMICAL⟩ effect on production ⟨CHEMICAL⟩
- ⟨CHEMICAL⟩ * of effects to * ⟨GENE⟩
  ⟨CHEMICAL⟩ exposure associated with expression of gene ⟨GENE⟩
- ⟨CHEMICAL⟩ * apoptosis in lines * ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * through inhibition of signaling in cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * genes of expression induced in cells ⟨DISEASE⟩
  ⟨CHEMICAL⟩ * expression on effect * ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * genes of expression induced in cells * ⟨GENE⟩
  ⟨CHEMICAL⟩ * expression on effect * ⟨GENE⟩

Further lets take a look the top relations between our entities which are Chemical, Gene and Disease. The following are the top binary relations (disregarding syntactic generalization) retrieved for the type sets Chemical x Gene, Chemical x Disease, Gene x Disease. We have checked the existence of these relations by comparing with the relations in the knowledge base and noticed that all the binary relations shown below are present in the three knowledge bases Chemical-Gene, Gene-Disease and Chemical-Disease demonstrating the effectiveness of PATTY.

4.5.1 *Chemical x Gene.*

- ⟨CHEMICAL⟩ of effects on * ⟨GENE⟩
- ⟨CHEMICAL⟩ of effect on * ⟨GENE⟩
- ⟨CHEMICAL⟩ by inhibition of ⟨GENE⟩
- ⟨CHEMICAL⟩ had effect on * ⟨GENE⟩
- ⟨CHEMICAL⟩ by inhibition of * ⟨GENE⟩
- ⟨CHEMICAL⟩ by activation of ⟨GENE⟩
- ⟨CHEMICAL⟩ * expression of protein ⟨GENE⟩
- ⟨CHEMICAL⟩ increased expression of ⟨GENE⟩
- ⟨CHEMICAL⟩ by induction of ⟨GENE⟩
- ⟨CHEMICAL⟩ * had effect on * ⟨GENE⟩
- ⟨CHEMICAL⟩ treatment after increased * ⟨GENE⟩
- ⟨CHEMICAL⟩ of effects on ⟨GENE⟩
- ⟨CHEMICAL⟩ decreased expression of ⟨GENE⟩
- ⟨CHEMICAL⟩ by induction of * ⟨GENE⟩

Pattern	Confidence	Statistical Strength
⟨*DISEASE*⟩ of effects [WORD] * ⟨*DISEASE*⟩	0.002	1
⟨*CHEMICAL*⟩ ⟨*GENE*⟩ * ⟨*CHEMICAL*⟩	0.002	1
⟨*DISEASE*⟩ of effects ADP * ⟨*DISEASE*⟩	0.003	1
⟨*GENE*⟩ * ⟨*CHEMICAL*⟩ ⟨*DISEASE*⟩	0.003	1
⟨*DISEASE*⟩ * ⟨*GENE*⟩ ⟨*DISEASE*⟩	0.003	1
⟨*DISEASE*⟩ [WORD] effects on * ⟨*DISEASE*⟩	0.004	1
⟨*DISEASE*⟩ of effects on * ⟨*DISEASE*⟩	0.004	1
⟨*DISEASE*⟩ ADP effects on * ⟨*DISEASE*⟩	0.004	1
⟨*DISEASE*⟩ of effect [WORD] * ⟨*DISEASE*⟩	0.004	1
⟨*DISEASE*⟩ [WORD] effect on * ⟨*DISEASE*⟩	0.004	1

Table 11: Bottom 10 frequent patterns based on statistical strength for support threshold 20

Pattern	Confidence	Statistical Strength
⟨*CHEMICAL*⟩ * apoptosis ADP cells ⟨*DISEASE*⟩	0.542	114
⟨*CHEMICAL*⟩ * apoptosis [WORD] cells ⟨*DISEASE*⟩	0.542	114
⟨*CHEMICAL*⟩ ADP effects on * ⟨*DISEASE*⟩	0.385	91
⟨*CHEMICAL*⟩ of NOUN on * ⟨*DISEASE*⟩	0.243	91
⟨*CHEMICAL*⟩ of [WORD] on * ⟨*DISEASE*⟩	0.243	91
⟨*CHEMICAL*⟩ ADP effects on * ⟨*GENE*⟩	0.309	73
⟨*CHEMICAL*⟩ of effects on * ⟨*GENE*⟩	0.306	73
⟨*CHEMICAL*⟩ of NOUN on * ⟨*GENE*⟩	0.195	73
⟨*CHEMICAL*⟩ of [WORD] on * ⟨*GENE*⟩	0.195	73
⟨*CHEMICAL*⟩ ADP effect on * ⟨*GENE*⟩	0.460	65

Table 12: Top 10 frequent patterns based on statistical strength for support threshold 20

Pattern	Confidence	Statistical Strength
⟨*CHEMICAL*⟩ * PART NOUN ADP * ⟨*GENE*⟩	1.0	47
⟨*CHEMICAL*⟩ * on effect had ⟨*CHEMICAL*⟩	1.0	15
⟨*CHEMICAL*⟩ * ADP effect had ⟨*CHEMICAL*⟩	1.0	15
⟨*CHEMICAL*⟩ * [WORD] effect had ⟨*CHEMICAL*⟩	1.0	15
⟨*CHEMICAL*⟩ * on NOUN had ⟨*CHEMICAL*⟩	1.0	15
⟨*CHEMICAL*⟩ * on [WORD] had ⟨*CHEMICAL*⟩	1.0	15
⟨*CHEMICAL*⟩ * on effect VERB ⟨*CHEMICAL*⟩	1.0	15
⟨*CHEMICAL*⟩ PART exposure after * ⟨*GENE*⟩	1.0	14
⟨*CHEMICAL*⟩ induced expression of ⟨*GENE*⟩	1.0	10
⟨*CHEMICAL*⟩ induced NOUN of ⟨*GENE*⟩	1.0	10

Table 13: Top 10 frequent patterns based on confidence for support threshold 20

- ⟨CHEMICAL⟩ treatment led to * ⟨GENE⟩
- ⟨CHEMICAL⟩ * in role of ⟨GENE⟩
- ⟨CHEMICAL⟩ to exposure of * ⟨GENE⟩
- ⟨CHEMICAL⟩ * activation of pathway ⟨GENE⟩
- ⟨CHEMICAL⟩ * inhibition of activity ⟨GENE⟩
- ⟨CHEMICAL⟩ with treated cells * ⟨GENE⟩

4.5.2 *Chemical x Disease.*

- ⟨CHEMICAL⟩ of effects on ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * growth of cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * in lines cell ⟨DISEASE⟩
- ⟨CHEMICAL⟩ of effects in * ⟨DISEASE⟩
- ⟨CHEMICAL⟩ of effect in * ⟨DISEASE⟩
- ⟨CHEMICAL⟩ by induced in * ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * proliferation of cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ of effect on ⟨DISEASE⟩
- ⟨CHEMICAL⟩ of injection by * ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * in treatment of ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * for treatment of ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * activity in cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * in line cell ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * effects on cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * in model of ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * arrest in cells ⟨DISEASE⟩

- ⟨CHEMICAL⟩ * to response in * ⟨DISEASE⟩
- ⟨CHEMICAL⟩ * activity against cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ resistance in cells ⟨DISEASE⟩
- ⟨CHEMICAL⟩ of effects in ⟨DISEASE⟩

4.5.3 *Gene x Disease.*

- ⟨GENE⟩ * apoptosis in cells ⟨DISEASE⟩
- ⟨GENE⟩ expression in cells ⟨DISEASE⟩
- ⟨GENE⟩ * in lines cell ⟨DISEASE⟩
- ⟨GENE⟩ of expression in * ⟨DISEASE⟩
- ⟨GENE⟩ * activation in cells ⟨DISEASE⟩
- ⟨GENE⟩ of role in ⟨DISEASE⟩
- ⟨GENE⟩ * in line cell ⟨DISEASE⟩
- ⟨GENE⟩ * growth of cells ⟨DISEASE⟩
- ⟨GENE⟩ plays role in * ⟨DISEASE⟩
- ⟨GENE⟩ of role in * ⟨DISEASE⟩
- ⟨GENE⟩ expression of cells ⟨DISEASE⟩
- ⟨GENE⟩ of expression in ⟨DISEASE⟩
- ⟨GENE⟩ * increased in cells ⟨DISEASE⟩
- ⟨GENE⟩ * of effects in * ⟨DISEASE⟩
- ⟨GENE⟩ * levels in cells ⟨DISEASE⟩
- ⟨GENE⟩ pathway in cells ⟨DISEASE⟩
- ⟨GENE⟩ levels in cells ⟨DISEASE⟩
- ⟨GENE⟩ apoptosis in cells ⟨DISEASE⟩
- ⟨GENE⟩ * in pathogenesis of ⟨DISEASE⟩

5 CONCLUSION

The PATTY algorithm is implemented on a text corpus related to Chemical, Gene and Disease based entities. The algorithm has been run with different support thresholds for n-gram mining and the patter synsets were extracted. We were able to observe some significant differences based on the increase in threshold. For Support 5, the algorithm generated 60K pattern synsets on the text corpus. With some constraints in generalization it is still able to extract useful relations from unstructured corpus. We also observed that performance improvement by using syntactic generalization will be better observed in very large corpus rather than smaller dataset. We speculate that maybe using a very large dataset like the author of PATTY paper used (YAGO dataset and Wikipedia corpus) does not lead to such issues.

REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. 1993. Mining association rules between sets of items in large databases. In *Acm sigmod record*, Vol. 22. ACM, 207–216.

[2] Explosion AI. 2017. Industrial-Strength Natural Language Processing. Retrieved 2017 from https://spacy.io/

[3] NCSU Biological Laboratory. 2018. Comparative Toxicogenomics Database. Retrieved November, 2018 from http://ctdbase.org/downloads/

[4] Marie-Catherine De Marneffe, Bill MacCartney, Christopher D Manning, et al. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC*, Vol. 6. Genoa Italy, 449–454.

[5] Patrick Ernst, Amy Siu, and Gerhard Weikum. 2018. HighLife: Higher-arity Fact Harvesting. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1013–1022.

[6] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *ACM sigmod record*, Vol. 29. ACM, 1–12.

[7] Johannes Hoffart, Fabian M Suchanek, Klaus Berberich, Edwin Lewis-Kelham, Gerard De Melo, and Gerhard Weikum. 2011. YAGO2: exploring and querying world knowledge in time, space, context, and many languages. In *Proceedings of the 20th international conference companion on World wide web*. ACM, 229–232.

[8] Girija Limaye, Sunita Sarawagi, and Soumen Chakrabarti. 2010. Annotating and searching web tables using entities, types and relationships. *Proceedings of the VLDB Endowment* 3, 1-2 (2010), 1338–1347.

[9] Thahir P Mohamed, Estevam R Hruschka Jr, and Tom M Mitchell. 2011. Discovering relations between noun categories. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 1447–1455.

[10] Andrea Moro and Roberto Navigli. 2013. Integrating Syntactic and Semantic Analysis into the Open Information Extraction Paradigm.. In *IJCAI*. 2148–2154.

[11] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2012. PATTY: a taxonomy of relational patterns with semantic types. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, 1135–1145.

[12] Ndapandula Nakashole, Gerhard Weikum, and Fabian Suchanek. 2013. Discovering semantic relations from the web and organizing them with patty. *ACM SIGMOD Record* 42, 2 (2013), 29–34.

[13] Ndapandula T Nakashole. 2012. Automatic extraction of facts, relations, and entities for web-scale knowledge base population. (2012).

[14] Roberto Navigli and Simone Paolo Ponzetto. 2010. BabelNet: Building a very large multilingual semantic network. In *Proceedings of the 48th annual meeting of the association for computational linguistics*. Association for Computational Linguistics, 216–225.

[15] Simone Paolo Ponzetto and Michael Strube. 2007. Deriving a large scale taxonomy from Wikipedia. In *AAAI*, Vol. 7. 1440–1445.

[16] Jun Zhu, Zaiqing Nie, Xiaojiang Liu, Bo Zhang, and Ji-Rong Wen. 2009. StatSnowball: a statistical approach to extracting entity relationships. In *Proceedings of the 18th international conference on World wide web*. ACM, 101–110.