NgramQuery - Smart Information Extraction from Google N-gram using External Resources

Martin Aleksandrov[†] **and Carlo Strapparava**[⋆]

[†]Sofia University, Sofia, Bulgaria
martina@fmi.uni-sofia.bg

[⋆]FBK-irst, Trento, Italy
strappa@fbk.eu

Abstract

This paper describes the implementation of a generalized query language on Google Ngram database. This language allows for very expressive queries that exploit semantic similarity acquired both from corpora (e.g. *LSA*) and from *WordNet*, and phonetic similarity available from the *CMU Pronouncing Dictionary*. It contains a large number of new operators, which combined in a proper query can help users to extract n-grams having similarly close syntactic and semantic relational properties. We also characterize the operators with respect to their corpus affiliation and their functionality. The query syntax is considered next given in terms of Backus-Naur rules followed by a few interesting examples of how the tool can be used. We also describe the command-line arguments the user could input comparing them with the ones for retrieving n-grams through the interface of Google Ngram database. Finally we discuss possible improvements on the extraction process and some relevant query completeness issues.

Keywords: N-grams, WordNet, Lexical Similarity

1. Introduction

In computational linguistics many tasks are properly dealt exploiting a combination of syntagmatic and domain features. Syntagmatic aspects are often managed by taking advantage of the analysis of large n-gram databases, while domain and ontological aspects are more properly modeled by semantic similarity spaces (e.g. latent semantic space) and lexical ontologies such as WordNet.

This paper describes the implementation of a generalized query language on Google Ngram database, whose expressiveness boosting is accomplished by plugging semantic similarity acquired both from corpora (e.g. LSA) and from WordNet, and phonetic similarity available from pronouncing dictionaries. It contains several different operators, which combined in a proper query can help users to extract n-grams having similarly close syntactic and semantic relational properties. We also characterize the operators with respect to their functionality, i.e. similarity, part-of-speech, syntactic and semantic operators.

The tool can be useful in a variety of tasks, ranging from specific lexicon extraction and lexical substitution task (McCarthy and Navigli, 2007) to the automatization of creative and figurative language processes (Strapparava et al., 2007; Stock et al., 2008; Veale, 2011) such as puns and portmanteau words generation, metaphors, hyperbolae and other rhetorical phenomena.

The paper is organized as follows. Section 2. briefly presents the notions of what we call *The Core language* and *The Ngram-query language*. Section 3. describes the operators we have implemented and it groups them into categories according to their functionality. In Section 4. we present how we can extract domain-specific concepts and how we can deal with orthographic and phonetic rhymes. Finally, in Section 5. we describe the expressiveness of the tool and some promising future directions.

2. Implementation

2.1. Preliminary notes

The starting point was the Google Web 1T 5-Grams database (Brants and Franz, 2006). This data set contains English word n-grams and their observed frequency counts. The length of the n-grams ranges from unigrams to five-grams. The n-gram counts were generated from approximately 1 trillion word tokens of text from publicly accessible Web pages. For easily handling that huge dataset, we exploited Web1T5-Easy[1], which is a collection of Perl scripts for indexing and querying the Google Web 1T 5-Grams database with the open-source database engine SQLite. The creation of our tool was inspired by the idea of combining different knowledge sources, in particular:

- Lexical concepts and their taxonomies, given by *WordNet* lexicon database. Besides all the relations present in WordNet, we embedded in the query language also the common similarity measures (Budanitsky and Hirst, 2006; Pedersen et al., 2004) such as Resnik, Lin, Jiang-Conrath, etc.
- Exploiting a specific version of *LSA* space (Deerwester et al., 1990) acquired from the full British National Corpus.
- The *CMU Pronouncing Dictionary*[2], for dealing with assonances, partial homophones, etc.

Finally we designed a query language that is able to express these concepts according Google N-gram database. Information extraction from such a database using fixed external

[1]http://cogsci.uni-osnabrueck.de/∼korpora/ws/cgi-bin/Web1T5/Web1T5_freq.perl

[2]http://www.speech.cs.cmu.edu/cgi-bin/cmudict

resources is a difficult task, especially with respect to efficiency and completeness. We will discuss some of these issues in the Section 5.
To give a flavor of the expressiveness, a possible query could be:

$$\sim\#food\ sandwich\ with\ cheese\#L\#10$$

that retrieves the 4-grams, along with their frequencies, in which the first word is a hyponym of $food$ (i.e. the term $\sim\#food$), then '`sandwich with`' followed by one of the first 10 similar words to *cheese* according to LSA (i.e. *cheese*$\#L\#10$). The most frequent 4-gram (with frequency 896) that satisfies the query is:

```
896 egg sandwich with tomato
```

2.2. Core language

Web1T5-Easy offers a quick and convenient way to build an interactively searchable version of the Web1T5 database, including a full collocation analysis and it provides five search terms, i.e. *a literal term*, *a word set term*, *wildcard term* (use of %), *arbitrary word term* (asterisk *) and *skipped token term* (question mark ?). They give to the user a simple extraction power. Nonetheless, they are quite useful and we built our extension on top of them. We call the set of queries constructed using these terms *The Core-query language*.

2.3. Generalized language

We call *The Ngram-query language* the extension of the basic query language. It contains a large number of new operators, with proper syntax and semantics. They allow the user to combine syntagmatic and domain operators within a single query and to retrieve resource-dependent n-grams in a fast and efficient manner.

3. Operators

In this section we present the operators implemented by dividing them into groups. This division was inspired by having different external corpora, i.e. WordNet lexicon database, a specific version of LSA space and the *CMU Pronouncing Dictionary*, on one hand and the operator functionality, i.e. pointer-concept, similarity and relatedness, part-of-speech, syntactic and phonetic operators, on the other.

3.1. WordNet

3.1.1. Relational operators

The organization of the lexical knowledge in the WordNet repository is represented by a variety of lexical and semantic relations. While lexical relations holds between word forms, the semantic relations hold between word meanings (i.e. *synsets*). We used pointers to represent the relations between the words in one synset and another. Lexical pointers relate specific word forms in the source and taget synsets. Semantic pointers relate word meanings, and therefore pertain to all of the members in the source and target synsets. In addition, the repository files lexicographically correspond to one of the four syntactic categories, i.e. noun, verb, adjective and adverb.

In Table 1 the operators implemented for each of the WordNet categories are listed.

Operator	**n**	**v**	**a**	**r**
Antonym (!)	+	+	+	+
Hypernym (@)	+	+	-	-
Instance hypernym ($@i$)	+	-	-	-
Hyponym ($\sim$)	+	+	-	-
Instance hyponym ($\sim i$)	+	-	-	-
Member holonym ($\#m$)	+	-	-	-
Substance holonym ($\#s$)	+	-	-	-
Part holonym ($\#p$)	+	-	-	-
Member meronym ($\%m$)	+	-	-	-
Substance meronym ($\%s$)	+	-	-	-
Part meronym ($\%p$)	+	-	-	-
Attribute ($=$)	+	-	+	-
Derivationally related form ($+$)	+	+	-	-
Entailment ($*$)	-	+	-	-
Cause ($>$)	-	+	-	-
Also see (^)	-	+	+	-
Verb group ($)	-	+	-	-
Similar to (&)	-	-	+	-
Participle of verb ($<$)	-	-	+	-
Pertainym (\\)	-	-	+	-
Derived from adjective (\\)	-	-	-	+

Table 1: Pointer-concept operators. Abbreviation are as follows: **n** (*noun*); **v** (*verb*); **a** (*adjective*); **r** (*adverb*);
$+$ means the relation is present in WN, and $-$ otherwise.

To clarify the use of the pointer operators we provide some examples. First, let us consider the query:

$$\sim\#food\#n$$

that retrieves the hyponyms of the noun $food$ for all of its senses. Some of the returned unigrams, along with their frequencies in the Google Web 1T5-gram database, are given below.

```
...
21757873 dish
20962063 milk
...
17654214 meat
16705545 cheese
16529672 cake
15533365 fiber
...
11444465 beef
10977879 juice
10888394 eggs
...
9757346 salmon
...
```

Our second example aims at uncovering the expressiveness of the new language:

$$\sim\#query\#v$$

which extracts the hyponyms of the verb $query$. Possible concepts with their frequencies are listed next.

```
...
130648714 question
...
22807901 wonder
18645408 pump
13032063 examine
11868500 quiz
...
5968768 inquire
2653183 enquire
155855 debrief
...
```

We also give a third example this time requiring the hypernyms of the $vehicle$ for all of its senses.

$$@\#vehicle$$

```
...
47047997 transport
16188678 substance
5083861 instrumentation
1133899 conveyance
...
```

3.1.2. Similarity operators

Exploiting the topology and the content of the WordNet repository it is possible to implement a variety of similarity and relatedness measures as operators in the extended language. The authors in (Pedersen et al., 2004) present a simple and useful Perl-implemented software package, Wordnet::Similarity, which can be used for retrieving information related with the degree of similarity and relatedness between two concepts either within a single part of speech space or between different syntactic category spaces. In the former case they discuss 6 similarity measures, i.e. 3 *LCS*-based and 3 path-based, which are followed by a short presentation of 3 relatedness measures possibly crossing the boundaries of the part of speech spaces. We used the provided API to include all 9 measures as operators in the *Ngram-query language*. Furthermore, we added the possibility for the user to set a random measure, which chooses in a stochastic manner one of the other measures, and a similarity or relatedness threshold, i.e. if two concepts have measure value below this threshold they are discarded. Table 2 presents these operators together with their threshold.

Operator	**acr**	**t**	**sim**	**rel**
Resnik	res	≥ 1.37	+	-
Lin	lin	$[0, 1]$	+	-
Jiang & Conrath	jcn	≥ 0	+	-
Leac. & Chod.	lch	$[0, 3.68)$	+	-
Wu & Palmer	wup	$(0, 1]$	+	-
Measure path	path	$(0, 1]$	+	-
Hirst & St-Onge	hso	$[0, 16]$	-	+
Baner. & Peder.	lesk	≥ 1	-	+
Patwardhan	vector	$[0, 1]$	-	+
Random	random	above	+	+

Table 2: WordNet measures operators. Abbreviation are as follows: **acr** (*similiarity measure acronym*); **t** (*threshold range*); **sim** (*similarity*); **rel** (*relatedness*).

3.2. LSA

In addition to similarity measures obtained using lexical resources such as WordNet, we intended to exploit in the query language also a corpus-based semantic similarity mechanism. As a corpus-based measure of semantic similarity we exploited latent semantic analysis (LSA) proposed by (Landauer et al., 1998). In LSA, term co-occurrences in a corpus are captured by means of a dimensionality reduction operated by a singular value decomposition (SVD) on the term-by-document matrix representing the corpus. For the implementation reported in this paper, we run the SVD operation on the the full British National Corpus, using 400 dimensions. Then for each word in the corpus, we cached the first hundred most similar words to that word, with the respective *cosine* values.

We embedded the LSA similarity into the query syntax in two settings: a threshold for the number of similar words, and a threshold for the *cosine* value between words. The user has the freedom to insert them individually or together within a single query term. As an example, the query

$$vehicle\#L\#0.7$$

```
57242495 traffic
48465416 driver
...
15093824 roads
9322146 brake
3434989 pedestrian
...
617922 motorist
...
```

gives the concepts which are similar to the noun $vehicle$ and have cosine value greater or equal than 0.7. The second example generalizes the settings by specifying the number of the required LSA terms within the query in addition to their cosine. It retrieves the similar terms of $food$ with cosine value at least 0.6 among the first 15 most similar terms.

$$food\#L\#15\#0.6$$

```
44677719 diet
...
21566476 eating
...
2989025 poisoning
...
896822 nutritious
249796 sugary
...
```

3.3. Phonetic dictionary

The *Carnegie Mellon University* Pronouncing Dictionary is a machine-readable pronunciation dictionary, which contains over a hundred twenty-five thousands words and their translations. The last is built on top of the phoneme domain, which currently contain 39 phonemes, for which the vowels may carry a flag indicating their lexical stress, i.e. 0 for no stress, 1 for primary stress and 2 for secondary stress. We implemented a single operator, which allows us to retrieve words with similar phonemic beginnings, endings or both. Its acronym is £ and it is applied in a manner similar to the orthographic substring matching % operator in *The Core-query language*, i.e. `£tion`, `mod£` or `£gmat£`. In Table 3 a comparison between the two operators is given.

Operator	**acr**	**begin**	**middle**	**end**
Orthographic	%	+	+	+
Phonetic	£	+	-	+

Table 3: Syntactic and phonetic operators. Operator applicability: **begin** (*at the beginning of a word*); **middle** (*within a word*); **end** (*at the end of a word*)

The orthographic operator can be applied multiple times within a word, while the phonetic one can only be applied either at the beginning or the end of a word (or both). The following two examples clarify the use of the two operators. They also show how the user can use the *word set term* (i.e. asking for two different n-grams in the same query).

$$\%quer\%, \%icle$$

```
181971412 article
 54080132 vehicle
 31285263 query
      ...
  8329460 particle
  7821136 albuquerque
  6939347 chronicle
      ...
   661990 konqueror
   563796 follicle
   439169 ventricle
      ...
```

We can see that % can be applied more than one times within a string. This is not the case with the operator £ as shown next. Recall, when £ operator is applied to a string then an overlapping between the given string and the words in the phonetic dictionary is performed and the positive matches are returned as part of the results to the query.

$$\pounds fore$$

```
5933321709 for
 277552584 before
 129170208 four
   2380057 fore
    542725 heretofore
       ...
```

3.4. Query syntax

In this subsection we give a formalization of the query syntax in *The Ngram-query language* in terms of Backus-Naur grammar rules.

3.4.1. Grammar

Note that we have a natural restriction on the length of the query imposed by the fact that Google Ngram database contains up to five grams.

⟨*query*⟩ ::= '`<term>`' *;; unigram*
| '`<term> <term>`' *;; bi-grams*
| '`<term> <term> <term>`' *;; 3-grams*
| '`<term> <term> <term> <term>`' *;; 4-grams*
| '`<term> <term> <term> <term> <term>`'

A query is composed of terms. The terms, in particular, can be simple words or a combination of words and operators. We clear the details with the following rules presenting the syntax of the different kinds of terms.

⟨*term*⟩ ::= ⟨*term_0*⟩ | ⟨*term_1*⟩ | ⟨*term_2*⟩ | ⟨*term_3*⟩

⟨*term_0*⟩ ::= ⟨*expr*⟩

⟨*expr*⟩ ::= ⟨*word*⟩ | ⟨*%-string*⟩ | ⟨*£-string*⟩
| ⟨*string-£*⟩ | ⟨*expr*⟩,(⟨*expr*⟩ , . . .)

⟨*%-string*⟩ ::= ⟨*%_within_concept*⟩

⟨*£-string*⟩ ::= ⟨*£_string*⟩

⟨*string-£*⟩ ::= ⟨*string_£*⟩

Examples of possible expressions are: `'house'`, `'%ing'`, `'£ture'`, `'ref£'`, `'data, roof, ref£, %mics, ad%nt%e, £ics'`.

⟨*term_1*⟩ ::= ⟨*p_c*⟩#⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩
| ⟨*term_0*⟩#⟨*L*⟩#⟨*sim*⟩#⟨*freq*⟩

⟨*p_c*⟩ ::= pointer_concept

⟨*pos*⟩ ::= n | v | a | r

⟨*s_n*⟩ ::= sense_number

⟨*L*⟩ ::= letter_L

⟨*sim*⟩ ::= number_similar_words

⟨*freq*⟩ ::= frequency_threshold

In the terms, following the first case of this pattern, part-of-speech (⟨*pos*⟩) and the sense number (⟨*s_n*⟩) settings can be omitted. Instead their defaults values are used, i.e. noun and all senses, respectively. In the second case at least one setting must be specified, i.e. either the number of similar concepts or the cosine threshold. Note that the number of similar words should be less than one hundred, a constraint imposed by the specificity of the LSA matrix we used (see Section 3.2.). The following examples give some possible terms according to the above pattern, i.e. `'@#fast#a#2'`, `'concept#L#10#0.7'`, `'data#L#0.3'`, `'~term#1'`.

⟨*term_2*⟩ ::= ⟨*p_c*⟩#⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩
#⟨*L*⟩#⟨*sim*⟩#⟨*freq*⟩

| ⟨*p_c*⟩#⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩#⟨*C*⟩#⟨*concept*⟩

| ⟨*term_0*⟩#⟨*L*⟩#⟨*sim*⟩#⟨*freq*⟩#⟨*C*⟩#⟨*concept*⟩

| ⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩#⟨*W*⟩#⟨*type*⟩#⟨*freq*⟩#⟨*s_n*⟩
#⟨*C*⟩#⟨*concept*⟩

⟨*W*⟩ ::= letter_W

⟨*type*⟩ ::= Wordnet_similarity_measure

⟨*C*⟩ ::= letter_C

⟨*term_3*⟩ ::= ⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩#⟨*L*⟩#⟨*sim*⟩#⟨*freq*⟩
#⟨*W*⟩#⟨*type*⟩#⟨*freq*⟩#⟨*s_n*⟩#⟨*C*⟩#⟨*concept*⟩

| ⟨*p_c*⟩#⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩#⟨*L*⟩#⟨*sim*⟩#⟨*freq*⟩
#⟨*C*⟩#⟨*concept*⟩

| ⟨*p_c*⟩#⟨*term_0*⟩#⟨*pos*⟩#⟨*s_n*⟩
#⟨*W*⟩#⟨*type*⟩#⟨*freq*⟩#⟨*s_n*⟩#⟨*C*⟩#⟨*concept*⟩

3.4.2. Command Line

The user can use the query language using the terminal interface. A possible query is

perl NgramQuery.perl -o -l 50 -f 1000 -d 3 '∼ #food'

First the query retrieves the hyponyms of '$food$', which are down to the third level in the corresponding *is-a* hierarchy of WordNet, then the first 50 unigrams with frequency at least 1000 are proposed.

4. Examples

This section shows some examples on how to use the tool and how it is possible to combine different resources within a query.

4.1. Example 1: Automated creation of fast-food menu

The query:

$\sim\#food, drink\ \ sandwich, pizza, pasta, salad\#L\#10$

will retrieve bigrams, along with their frequencies, in which the first component is an hyponym of the word $food$ or $drink$ and the second one is a word within the first 10 LSA most similar words to either $sandwich$, $pizza$, $pasta$ or $salad$. In such a way we can retrieve all the combinations not only correct with respect to the WordNet ontology and LSA similarity, but also reasonable with respect to Google Ngram database, i.e. to the way people use them. For instance, `egg burger`, `tomato salad`, `cheese spaghetti`, etc. are foods people used to eat, while `chocolate gringer`, `sugar lasagna` are not.

4.2. Example 2: Interesting activities

The query:

$interesting\ \ \%ing\#n\#1\#W\#res\#2.1\#1\#C\#activity$

retrieves all bigrams in which the first word is `interesting` followed by a word, ending in $-ing$, which has to be a noun and its first sense in WordNet has to be similar, according to Resnik similarity measure with threshold ≥ 2.1, to the first sense of $activity$. The first two results are:

```
3679 interesting shopping
2409 interesting training
             ...
```

5. Future work and discussion

In the first example above the reader can observe that the query contains two terms and the results are bigrams. What about multiple-word concepts such as *modern* and *state-of-the-art*? The user might want to retrieve *state-of-the-art technique* when input the query $\mathcal{Q}$ =*∼#modern approach#L#20*. We will address such a situation as a future improvement.
An important issue is the trade-off between the expressiveness of the query language and the complexity of the retrieval. Imagine the user wants to retrieve 4-grams by entering the following query, $\mathcal{Q}$ = *fast#a#L#10 old#a#1#L#10#W#lin#0.2#C#archaic &#cheap#L#0.8 vehicle#L#10*. How complex is answering to this query? A complete freedom in combining operators leads quickly to computational tractability problems. At present this issue is addressed by relying on user awareness of keeping the queries sensible. We are planning to implement a preventive analysis of the query, providing the user with precise help for faster and more reliable retrieving.

6. Conclusions

We developed a generalized query language on Google Ngram database, whose expressiveness is boosted by plugging semantic similarity acquired both from corpora (e.g. LSA) and from *WordNet*, and phonetic similarity available from pronouncing dictionaries, in particular *CMU Pronouncing Dictionary*. The language implements a large number of new operators, each one of them with different meaning and purpose. We summarized them and clarified their syntax through usage examples. Then, we presented the syntax of the new language and discussed the settings involved. We think that this tool will be helpful in many tasks some of which are specific lexicon extraction, creative language combinations, and generation of language variations.

Acknowledgements

Martin Aleksandrov was partially supported by the European Master Program in Computational Logic (EMCL).
Carlo Strapparava was partially supported by a Google Research Award.
Finally, we would like to thank Marco Guerini and Gözde Özbal for many useful and valuable discussions.

7. References

T. Brants and A. Franz. 2006. Web 1T 5-gram version 1. Linguistic Data Consortium.

A. Budanitsky and G. Hirst. 2006. Evaluating WordNet-based measures of lexical semantic relatedness. *Computational Linguistics*, 32:13–47.

S. Deerwester, S. T. Dumais, G. W. Furnas, T.K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41:391–407.

T. K. Landauer, P. Foltz, and D. Laham. 1998. Introduction to latent semantic analysis. *Discourse Processes*, 25.

D. McCarthy and R. Navigli. 2007. The semeval English lexical substitution task. In *Proceedings of the ACL Semeval workshop*.

T. Pedersen, S. Patwardhan, and J. Michelizzi. 2004. WordNet::Similarity - measuring the relatedness of concepts. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-2004)*, pages 1024–1025, San Jose, CA, July. Lawrence Erlbaum Associates.

O. Stock, C. Strapparava, and A. Valitutti. 2008. Ironic expressions and moving words. *International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI)*, 22:1045–1057.

C. Strapparava, A. Valitutti, and O. Stock. 2007. Affective text variation and animation for dynamic advertisement. In *Proccedings of 2nd International Conference on Affective Computing and Intelligent Interaction (ACII2007)*, Lisbon, Portugal, September.

Tony Veale. 2011. Creative language retrieval: A robust hybrid of information retrieval and linguistic creativity. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 278–287, Portland, Oregon, USA, June. Association for Computational Linguistics.