A Cohesion-based Approach for Unsupervised Recognition of Literal and Nonliteral Use of Multiword Expression

by

Linlin Li

Submitted to the Department of Computational Linguistics
in partial fulfillment of the requirements for the degree of

Master of Science in Computational Linguistics

at the

UNIVERSITÄT DES SAARLANDES

November 2008

Author . . . .
Department of Computational Linguistics
November 11, 2008

Certified by . . . .
Caroline Sporleder
Dr.
Thesis Supervisor

Certified by . . . .
Manfred Pinkal
Prof.
Thesis Supervisor

Accepted by . . . .
Werner Saurer
Chairman, Department Committee on Graduate Students

A Cohesion-based Approach for Unsupervised Recognition of Literal and Nonliteral Use of Multiword Expression

by

Linlin Li

Submitted to the Department of Computational Linguistics
on November 11, 2008, in partial fulfillment of the
requirements for the degree of
Master of Science in Computational Linguistics

Abstract

Texts frequently contain expression whose meaning is not strictly literal, such as idioms. Idiomatic and non-literal expressions pose a major challenge to natural language processing technology as they often exhibit lexical and syntactic idiosyncrasies. We propose a novel unsupervised method for distinguishing literal and non-literal usages of expressions. Our method determines how well a literal interpretation of the expression is linked to the overall cohesive structure of the discourse. If only weak cohesive links can be found, the expression is classified as idiomatic. We propose two methods to model the cohesive links in our task: the lexical-chain-based approach and the cohesion-graph-based approach. While the chain-based approach is effective at distinguishing literal and non-literal usage, it is sensitive to chaining algorithms, parameter settings and data setup. We further develop the chain-based approach into a graph-based approach in order to overcome these problems. This development makes our cohesion-based approach unsupervised while maintaining a high performance.

Thesis Supervisor: Caroline Sporleder
Title: Dr.

Thesis Supervisor: Manfred Pinkal
Title: Prof.

Acknowledgments

There is no doubt that my thesis supervisor, *Dr. Caroline Sporleder*, deserves the first place on this list. She gives me so much valuable suggestions during the whole time of the work. Without her help, I would have got struck somewhere in the middle not knowing where to go, or lost in the detail forgetting about the general direction. She is a nice person and a great supervisor!

I would like to thank my second supervisor, *Prof. Manfred Pinkal*, who offers me the opportunity to work in the SALSA research group where I have learned really a lot, for his constructive comments.

Many thanks to *Yi Zhang, Will Roberts and Daniel Bauer* for their help and feedback.

Thanks to *Prof. Hans Uszkoreit* for introducing me this master program, and offering me the opportunity to discover so many interesting things in another culture.

Great gratitude to my parents, *Xuejia Li* and *Zhiping Lin*, who have taught me the value of smile and trust, which makes most of things in life simpler and easier...

Many thanks for everybody who offered help, but I was so careless that I forgot to mention.

Contents

List of Tables

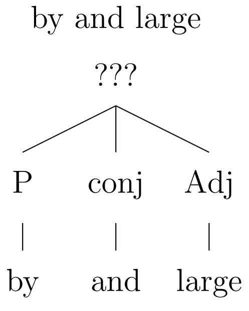
Chapter 1

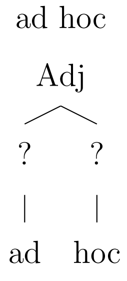
Introduction

1.1 Motivation

Multiword expressions (MWEs) are defined as "idiosyncratic interpretations that cross word boundaries or spaces"[32]. They are decomposable into multiple simplex words. Examples of MWEs are lexically fixed expressions (*e.g. ad hoc*), idioms (*e.g. see double*), light verb constructions (*e.g. make a mistake*) and institutionalized phrases (*e.g. kindle excitement*) [2]. MWEs are pervasive in natural language. They are estimated to be equivalent in number to simplex words in mental lexicon [17]. MWEs exhibit a number of lexical, syntactic, semantic, pragmatic and statistical idiosyncrasies:

- Syntactic idiosyncrasies

  The tag of the top node for the MWEs *by and large* cannot be decided with a standard syntactic parser, and the part-of-speech tags of the words *ad* and *hoc* are unclear.

- Semantic non-compositionality. There is often a mismatch between the semantics of the parts and the whole. For instance, the meaning of *kick the bucket* (die) and *red tape* (bureaucratic) cannot be easily predicted from the surface form of its components.

- Pragmatic idiosyncrasies. For some MWEs, the expression is associated with a fixed pragmatic point (*good morning, good night*).

- Variation in syntactic flexibility. There is considerable variation in syntactic flexibility between different expressions (I *handed in* my thesis = I *handed* my thesis *in*, Kim *kicked the bucket* $\neq$ **the bucket* was *kicked* by Kim).

- Variation in productivity. There are various levels of productivity for different MWEs (*kick/*beat/*hit the bucket*, *call/ring/phone/*telephone up*).

These idiosyncrasies pose challenges for NLP systems, which have to recognize that an expression is an MWE to deal with it properly. Recognizing MWEs has been shown to be useful for information retrieval [33, 22, 25, 41]. There have been some studies on the correlation of MWE identification and tagging accuracy [31] . It has also been shown that MWEs account for 8% of parsing errors with precision grammars [1]. MWEs are also used in infomation extraction [23], and are an integral component of symbolic MT systems [15, 6].

However, the special properties of MWEs can also be exploited to recognize MWEs automatically. There have been many studies: identification (determining whether multiple simplex words form a MWE in a given token context, e.g. *<u>put</u> the sweater <u>on</u>* vs. *<u>put</u> the sweater <u>on</u> the table*), extraction (recognizing MWEs as word units at the type level), detecting or measuring compositionality of MWEs, semantic interpretation (interpreting the semantic association among components in MWEs).

Various methods have been proposed to use the syntactic and lexical fixedness, or apply various statistical measures across all co-occurrence vectors between the whole expression and its component parts (see Section 1.2) for multiword extraction, identifying MWEs at a type level. These methods can be used to automatically identify potentially idiomatic expressions, but they do not say anything about the idiomaticity of an expression in a particular context. While some idioms (*ad hoc*) are always used idiomatically, there are numerous idioms that can be used both idiomatically (see Example 1.1) and non-idiomatically (see Example 1.2).

(1.1) When the members of De la Guarda aren't hanging around, they're yelling and *bouncing off the wall.*

(1.2) Blinded by the sun, Erstad leaped at the wall, but the ball *bounced off the wall* well below his glove.

Our work aims to identify the literal or non-literal usages of idiomatic expressions in a certain context. This is different from the type based classification, which classifies the idioms in a type level, i.e. distinguishing idiom phrases or non-idiom phrases irrespective of the context.

Our work is also different from semantic compositionality (see Section 1.2.2), as it studies the semantic compositionality of a given MWE in a certain context, which is not the same as the study of an MWE's semantic compositionality (irrespective of the context).

1.2 Related Work

1.2.1 Type-based Classification

Type-based classification aims to extract multiword expression types in text from observations of the token distribution. It aims to pick up on word combinations which occur with comparatively high frequencies when compared to the frequencies of the individual words. There are many different statistical tests for this, and the results from different studies conflict with one another, which may be caused by the fact that different distributional idiosyncrasies between different corpora, and different tests have different statistical idiosyncrasies.

Evert and Krenn [13] found that simple frequency is very good over German Adj-N and P-N-V triple extraction tasks. Smadja [36] uses an automatic method for extracting MWEs from raw text based on n-gram statistics. The idea is that MWEs are more rigid syntactically and more frequent than other word combinations. This method is not effective at extracting low-frequency words. Also, since the MWEs extraction is based on bigrams, it is difficult to extract non-contiguous MWEs such as verb particle constructions (VPC). Baldwin and Villavicencio [2] used a range of methods for shallow and deep lexical acquisition of verb-particle constructions from unannotated corpora. They implemented two rule based methods, utilizing the lexical and syntactic fixedness. They also used the lexical and syntactic fixedness as features for a machine learning classifier. The methods are robust over extremely low-frequency data.

1.2.2 MWE Semantic Compositionality

Another work that is similar to our work is the study of the semantic compositionality of MWEs. Semantic compositionality is the degree to which the semantics of the parts of an MWE contribute towards those of the whole. Semantic decomposability is the degree to which the semantics of an MWE can be ascribed to those of its parts. The realistic short-term objective of this task is to binary classify the MWEs as idiosyncratically decomposable (*e.g. spill the beans*) or non-decomposable (*e.g. kick the bucket*). Several approaches have been proposed:

Lin's [24] work attempts an automatic identification of non-compositional phrases. He used the substitution test[1] and mutual information to determine the compositionality of the phrase. The problem with using the substitution test for detecting non-compositionality is that some productive MWEs may have problems with this method. For example, *call up* and *ring up* are both MWEs. The substitution test would treat *ring up* as compositional since the word *ring* can be substituted for the word *call* in the substitution test, and the MI value of the phrase *ring up* would be close to *call up*, which is taken as the evidence of *ring up* being a compositional phrase.

[1]The substitution test aims to replace part of the idiom's words with semantically similar words, and test how the co-occurrence frequency changes.

A statistical approach to the semantics of verb-particles was proposed by Bannard et al. [3]. They defined distributional similarity, assuming that if an MWE is compositional, it will occur in the same lexical context as its component parts. The co-occurrence vector representations of verb partical construction (VPC) and the component words are used in different classification strategies.

1.2.3 Token-based Classification

There have also been a few token-based classification approaches, aimed at classifying individual instances of a potential idiom as literal or non-literal. Katz and Giesbrecht [19] make use of latent semantic analysis (LSA) to explore the local linguistic context that can serve to identify multi-word expressions that have non-compositional meaning.They measure the cosine vector similarity between the vectors associated with an MWE as a whole and the vectors associated with its constituent parts and interpret it as the degree to which the MWE is compositional. They report an average accuracy of 72%, but the data set used in their evaluation is small.

Birke and Sarkar [5] use literal and non-literal seed sets acquired without human supervision to perform bootstrapping learning. The new instances of potential idioms are always labeled according to the closest set. While their approach is unsupervised clustering, they do rely on some resources such as databases of idioms.

Fazly and Cook [14] developed statistical measures to measure the lexical and syntactic fixedness of a given expression, which is used to automatically recognize expression types, as well as their token identification in context. They report an average accuracy of 72% for their classifier.

1.3 Overview

The idea of this thesis is to make use of lexical cohesion (see Chapter 2) to identify idiomatic use of multiword expressions. The hypothesis is that while the component words of literal expressions fit into the lexical cohesion structure of the discourse, the non-literal expressions do not.

The first strategy described in Chapter 2 is to make use of lexical chains to model discourse cohesion, under the assumption that literal expressions participate in lexical chains, while non-literal expression do not. Results show that the idea is effective, but it has the disadvantage that the threshold for placing new words into existing chains is difficult to chose. The chain forming strategy also influences the performance. The experimental results of our study show that the proposed lexical chain based approach is very sensitive to differences in the threshold, chain forming algorithm , and experiment data settings.

In order to overcome these problems, we propose a new graph-based method in Chapter 3. Instead of looking into how literal expressions take part in a lexical chain, the graph-based idea is to find out how the literal expressions take part in the cohesion graph that models the whole discourse context. If the MWE expression is semantically well connected to the other words in the context, it is considered to be used literally. The advantage of this method is that it avoids the thresholds, chain forming strategy, and data setup problems. It also takes more cohesion information into account, since it tests the semantic connectivity of the literal expression to the remaining words in the context, instead of only considering limited number of words in existing chains. In other words, the cohesion graph models global cohesion while the lexical chain models local cohesion. Furthermore, it is totally unsupervised as no annotated data are necessary for parameter tuning. Experimental results show that the global lexical cohesion based graph method maintains as high performance as the supervised chain based approach.

Chapter 2

Lexical Chain-based Approach

A *lexical chain* is a cohesive chain in which the criterion for inclusion of a word is that it bears some kind of semantic relationship (not necessarily one specific relationship) to one or more words that are already in the chain [37]. Lexical chains can be computed based on the semantic relatedness of individual words, and they have been used successfully in many NLP applications: text summarization [4, 34, 35, 8, 12], word sense disambiguation [9, 28], text structure analysis [26] and malapropism correction [16].

Hirst and St-Onge [16] hypothesize that the more distant a word is semantically from all the other words of a text, the higher the probability is that it is a malapropism. A word in a text that cannot be fitted into a lexical chain, but which is close in spelling to a word that could be fitted, is likely to be a malapropism. In their experimental results, they show that actual malapropisms were 4.47 times more likely to be inserted in an atomic chain, which is a chain that only contains one word, meaning that the word does not fit into any other chains.

Much research has been done to use lexical chains for summarization based on the hypothesis that strong lexical chains are a good indicator of the topic of the text [8, 4, 12]. Brunn et al. [8], for example, choose segments that contain a large number of chain members, and then ranked each sentence by summing the number of shared chain members over the sentence. The most informative sentences are chosen as the summary. Barzilay and Elhadad [4] point out that given an appropriate measure of strength, picking the concepts represented by strong lexical chains gives a better indication of the central topic of a text than simply picking the most frequent words. Morris and Hirst [26] use thesaural relations to compute lexical chains, and determine the intentional structure of the text. Okumura and Honda [28] use lexical chains for text segmentation based on the idea that when a portion of a text forms a semantic unit, there is a tendency for related words to be used. The segmentation boundaries are selected in the order of boundary strength, which is calculated by the chain score of boundary words. Their experiment got an average recall rate of 52% and precision of 25%.

2.1 Modeling Semantic Relatedness

Our approach of token-based idiom classification depends crucially on the availability of a successful method for computing semantic relatedness. There are two main approaches. The lexical knowledge based approach relies on manually built lexical resources such as WordNet [39] or Roget's thesaurus [10], while the statistical based approach relies on the fact that semantically related words occur in similar context.

2.1.1 Related Work

Morris and Hirst [26] use Roget's thesaurus to model semantic relatedness. They implement five types of thesaural relations to define semantic relatedness. One word is thought to be semantically related to another if it is linked to the other word by one of these five types of thesaural relations with the other word. Their method also allows some degree of transitivity. Since they believe that two or more transitive links severely weaken the word relationship, they only use a transitivity of one link. Instead of using five thesaural relations, Jarmasz and Szpakowicz [18] define two relations according to the the word's index entries to identify semantic relatedness.

In addition to these thesaurus-based approach, there are also some approaches based on WordNet [37, 9, 8, 38]. In WordNet, a word may have more than one synset, each corresponding to a different sense of the word. When looking for a relation between two different words, St-Onge [37] considers the synsets of all the senses of each word that have not been ruled out, looking for a possible connection between some sense of the first and some sense of the second. He defines three kinds of relations:

- An extra-strong relation holds only between a word and its literal repetition
- Strong relations happen when there is a synset common to two words; or when there is a horizontal link (e.g., antonym) between a synset of each word
- A moderately strong relation between two words occurs when a member of a set of allowable paths connects a synset of each word. They define some rules to specify permissible paths, and the semantic relatedness proximity is calculated based on the weight of the path, which is in turn calculated based on the length and direction of the path.

Chali and Joty [9] create a hand-made table to define the semantic relatedness value based on the relations in WordNet, such as synonym, hypernym and hyponym, holonym and meronym etc. For instance, if one word is a synonym of the other word, the semantic relatedness of the two words is set to be 1, while if one word is a hypernym of the other, their semantic relatedness value is set to be 0.33.

Teich and Fankhauser [38] rely on the simplest type of lexical cohesion being repetition. The more complex types of lexical cohesion rely on the semantic relations between words, which are organized in terms of sense relations in WordNet.

One thing that has to be addressed about the WordNet-based approach is word sense disambiguation. How should a word be interpreted in a specific context? Which word senses in WordNet should be chosen?

Hirst and St-Onge [16] incorporate word sense disambiguation on the chain building algorithm. They construct a list of pointers to every synset of the word and attach them to the word. When a new chain starts, all its synsets are kept. Inserting another word into the chain results in a connection between the words by linking the synsets involved in the relation. When a word is inserted into a chain because of an extra-strong relation, all corresponding synsets are connected. When the relation involved is strong, all pairs of strongly related synsets are connected. When the relation involved is medium-strong, the pair of synsets whose weight is of the greatest weight is connected. After the connection between words is made, any unconnected synsets of the new word are deleted, and the chain is scanned to remove other synsets wherever possible.

Barzilay and Elhadad [4] point out that the greedy disambiguation strategy implemented in [16] has some limitations. They expained the limitation by an example (see Example 2.1).

(2.1) **Mr.** Kenny is the **person** that invented an anesthetic **machine** which uses **micro-computers** to control the rate at which an anesthetic is pumped into the blood. Such **machines** are nothing new. But his **device** uses two **micro-computers** to achieve much closer monitoring of the **pump** feeding the anesthetic into the patient.

The word "machine" was wrongly related to the chain "Mr., person", because the first WordNet sense of "machine" (an efficient person) is a holonym of "person". Even though later evidence supports the selection of "machine" over its common sense: "micro-computer", "device" and "pump", the wrongly-made decision cannot be corrected because the algorithm in [16] allows no way to backtrack.

In order to avoid the greedy decision problem, Barzilay and Elhadad propose that the whole picture of chain distribution in the text must be considered. They develop a chaining model according to all possible alternatives of word senses and then choose the best one among them. The best interpretation is defined as the one with the most connections. The score of an interpretation is calculated as the sum of its chain scores. The algorithm computes all possible interpretations, maintaining each one without self contradiction. When the number of possible interpretations is larger than a certain threshold, they prune the weak interpretations. In the end, the strongest interpretation is chosen.

Silber and McCoy [35] point out that Barzilay and Elhadad [4] implemented the algorithm in exponential time. They improved the algorithm by doing the same thing in linear time.

Chali and Joty [9] use a disambiguation graph where the nodes represent word instances with their WordNet senses and weighted edges connecting the senses of two different words represent semantic relation. They disambiguate the sense of a certain word by choosing the highest score sense node in the disambiguation graph.

In addition to the word sense disambiguation problem, the lexical knowledge dependent approaches have a lot more limitations:

Morris and Hirst [26] point out that the relationship between words is due more to their "feel" than their connectedness in a thesaurus. They give an example where the cohesive chain ***hand-in-hand***, **matching, whispering,** ***laughing***, ***warmth*** was not entirely computable. The words in italics were relatable by their properties of goodness, rather than by their specific meanings. Another example was the chain **environment, setting, surrounding** was not thesaurally relatable due to the fact that *setting* was not in the thesaurus, and although *environment* and *surrounding* are semantically related, they were not thesaurally connected. Furthermore, place names, street names, and people's names are generally not to be found in *Roget's Thesaurus.*

Using WordNet to compute semantic relatedness leads to similar problems: missing connections of certain semantically related words, inconsistency in the semantic proximity implicit in links in WordNet, incorrect or incomplete word sense disambiguation etc. [16]. Teich and Fankhauser [38] analyzed the problems with WordNet-based methods in more detail: missing relations (only some relations across part-of-speech are accounted for in WordNet), spurious relations (rather questionable ties are established without constraints on the length or branching factor of a transitive relation), sense proliferation (in some instances the sense-tagging appears to be overly specific. Using synonymy without repetition these senses does not form a link), and compound term problems (there might be some problem with sense-tagging with compound terms in WordNet).

2.1.2 Distributional Approach

For our task, we need to model a wide range of semantic relations. As there are many disadvantages with the knowledge dependent methods, we decided to use the distributional approach, choosing two statistically based models which compute relatedness on co-occurrence in a large corpus. The first method runs on the page count results returned by a search engine, using *Normalized Google Distance (NGD)* [11] to measure the association of two words. The second method is based on a standard distributional method [29], where the semantic relatedness value is calculated by the cosine value of the dependency vectors (DV) that representing each word. The fact that the former uses an online service, such as Yahoo, makes it utilize a much larger and more updated corpus than the latter which is relied on a fixed linguistic corpora.

Google distance is a measure of semantic relatedness derived from the number of page counts returned by a search engine for a given set of keywords. Keywords with the same or similar meaning tends to be "close" in the value of Google distance, while words with dissimilar meanings tend to be farther apart.

The normalized Google distance [11] between two search terms x and y is defined as Formula 2.2:

$$NGD(x, y) = \frac{\max\{\log f(x), \log f(y)\} - \log f(x, y)}{\log M - \min\{\log f(x), \log f(y)\}} \tag{2.2}$$

where M is the total number of web pages indexed by the search engine; $f(x)$ and $f(y)$ are the number of hits for search terms x and y respectively, and $f(x, y)$ is the number of page counts on which both x and y occur.

If the two search terms x and y never occur together, but do occur separately, the normalized Google distance between them is infinite, meaning they are not semantically related at all. In another extreme case where both terms always occur together, their NGD is zero, which is an indicator that they are semantically closely related.

While theoretically the scope of NGD is $[0, \infty)$, our experimental results show that most of the time the values fall between 0 and 1 (see [11] for a detailed discussion of the mathematical properties of NGD).

Dependency vector method is based on the traditional word-based co-occurrence models. Words are represented as vectors. Each dimension represents a chosen term. A word vector then contains the co-occurrence information between that word and the chosen terms. The hypothesis is that words that are semantically related are more likely to take similar context, thus the vector representations tend to be more similar. A variety of distance measures can be used to compute the similarity of two vectors; here, we use the method described by Pado and Lapata [29, 30], taking the cosine similarity which is define as:

$$sim_{cos}(\overrightarrow{x}, \overrightarrow{y}) = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.3}$$

We conducted a small-scale study in which we compared the semantic relatedness scores obtained by *NGD* and the *dependency vector* method to human ratings. Section 2.1.4 gives details of this evaluation.

2.1.3 Search Engine Stability

In order to test the **NGD-based Semantic Relatedness** approach, which is based on the results returned by the search engine API, we first carried out some experiments to test how stable the page counts returned by search engines are.

We used the two most widely-used search engines: Google[1] and Yahoo[2], and compared the performance of both.

There are several problems with the Search Engine API that may affect the performance of the NGD-based approach. The problems are problems shared by both search engines, but the example data shown here were based on Yahoo.

[1]Google Soap Search API was available at http://code.google.com/apis/soapsearch/. Although Google is no longer issuing new API keys for the SOAP Search API, they still support existing API keys. Fortunately, we have such a key.

[2]Yahoo Search Web Services as a Software Development Kit is available at http://developer.yahoo.com/download/download.html

	OPT = AND	*OPT* = OR
car	2,840,000,000	2,840,000,000
car *OPT* car	3,670,000,000	2,850,000,000
car *OPT* car *OPT* car	2,840,000,000	2,840,000,000
car *OPT* car *OPT* car *OPT* car	2,840,000,000	2,850,000,000

Table 2.1: Operator test with Yahoo

- The total number of the web pages indexed by a search engine is dynamic. The value M in Formula 2.2 is a function taking an argument of querying time T ($M(T) \neq constant$).

- The instability problem of the search engines[3]. The hits returned by the search engine vary at times (as shown in Example 2.4 and 2.5). This fact has to be modeled as the fact $N = f(t)$, where t is the query time, and N is the page count.

  (2.4) Hits(Jim) = 763,000,000
  Hits(Mary) = 748,000,000
  Hits(Jim AND Mary) = 103,000,000

  (2.5) Hits(Jim) = 757,000,000
  Hits(Mary) = 755,000,000
  Hits(Jim AND Mary) = 104,000,000

- The search engine AND and OR operators are unstable too(see Table 2.1). The values of the regular expressions *car, car OR car, car OR car OR car* are equal to each other, but the page counts returned by the search engine are not. Mathematically, the set indexed by *car AND car* is a subset of *car*, but our test queries show that it is the other way around.

- The page hits returned by the search engine's API is different from those re-

[3]Jean Veronis has worked a lot on this problem; see his blog, http://aixtal.blogspot.com/2005/02/web-googles-missing-pages-mystery.html

turned by its web interface (see Example 2.6, using keyword *foreign* as query).

(2.6) API: 1,020,000,000
Web: 1,040,000,000

- Both the Google and Yahoo API seemed to have problems with very high frequency words, with Google SOAP API throwing an exception as in Example 2.7 and Yahoo API retuning the number 2,147,483,647 for every high frequency word[4]).

(2.7) The call to the Google Web Service API failed:
com.google.soap.search.GoogleSearchFault: Fault Code = SOAP-ENV:Server Fault String = Exception from service object: For input string: "10980000000"

For the problems mentioned above, we propose the following solutions:

In the ideal case, if we find a function g which models the dynamic change of the total number of indexed web pages, where $M = g(T)$ represents the number of pages a search engine indexes in the specific time T, then the problem can be solved by replacing M with $g(T)$ in Formula 2.2. In reality, that is not the case. Search engines are always trying to index more web pages. We tried to find out an official number published by the search engine supplier, but no consistent result was found. As it is difficult to obtain a specific and reliable number for the number of pages indexed by a search engine, we simplified it as a constant M, and approximated it by setting it to the number of hits obtained for the word *the*. This assumes that the word *the* would occur in all English language pages [21].

The instability problem is even more sensitive to query time compared with the total indexed web pages problem. While the function $g(T)$ tends to be fairly constant over medium length time spans, the function $f(t)$ fluctuates much more. As the difference is not very significant over short time spans, we decided to take the page

[4]We think one possibility might be that it is the total number of indexed pages by the Yahoo API; another possibility might be that it is a data overflow problem caused by some technical reasons.

count N as a constant relative to query time t (as we have done with $g(T)$, $f(t)$ was treated as a constant N). We ignored the logical operator (*AND, OR*) instability problem also due to the fact that the difference is not significant.

In general, the variance in the number of pages indexed and searched is not such a big problem because it tends to be relatively stable over short time spans, and the querying in the experiment was done in one quick session without much delay.

We also ignored the fact that the result by the search engine API and web interface is inconsistent, due to the reason that these two numbers are almost linearly correlated, with the web number always a little higher than the API. This can be modeled as $w(x) = \alpha p(x)$, where $w(x)$ is the number of counts returned by web interface given term x, while $p(x)$ is the number of hits obtained by the search engine API with the same term. Since it is the relative frequency, $f(x)$, instead of the absolute count, $N(x)$, which is taken into consideration in Formula 2.2, decreasing the page number with a linear factor α by choosing the API number over the web interface does not affect the semantic relatedness calculation.

We excluded high frequency words based on the fact that a search engine exception can happen or the same 10-digit number can be returned. This amounted to filtering out function words.

When we implemented the algorithm, Yahoo was chosen over Google for the following reasons:

- Google is more inconsistent than Yahoo (as shown in Table 2.2). While nearly all the results by Yahoo API are a little lower than the web interface, the Google API tends to give more inconsistent results (the results by the web can be larger or smaller than the API). In general, we think Yahoo keeps more consistency between its development tool and the web interface by keeping a linear coefficient between them. We did not go into the mathematical proof of this problem due to the lack of a large amount of available data.

  It might be the case the Google utilizes a more complex distribution system than Yahoo. Different servers might return different page hits even at the same

	Google		Yahoo	
	Web	API	Web	API
magazine	509,000,000	428,000,000	2,030,000,000	1,940,000,000
girl	606,000,000	607,000,000	1,420,000,000	1,200,000,000
John	446,000,000	444,000,000	1,090,000,000	970,000,000

Table 2.2: Comparison with Google and Yahoo: page counts returned by the web interface (Web), page counts returned by the API (API)

querying time since it is difficult to maintain a complete synchronization in a large distributed system.

- From a technical point of view, the Google SOAP API ends the program by throwing an error with high frequency queries, which make subsequent execution impossible to continue. In contrast, it is easier to capture and handle Yahoo's unreliable 10-digit number while keeping the program running.

2.1.4 Comparison between NGD/semantic vector space and Human Rating

In order to test whether NGD is indeed correlated to semantic relatedness as rated by humans, we first ran the NGD-based approach on a German data set from the *Technische Universität Darmstadt* [5]. The data consists of a number of word pairs whose relatedness was judged by humans (on a scale of 0-4). The data set contains 3 subsets:

- **Gur65** contains 65 word pairs along with their relatedness scores assigned on a discrete 0-4 scale by 24 subjects. The inter-annotator agreement is 0.81.
- **Gur350** contains 250 word pairs along with their relatedness scores assigned on a discrete 0-4 scale by 8 subjects. The inter-annotator agreement is 0.69.

[5]These data are available from http://www.ukp.tu-darmstadt.de/data/semantic-relatedness/; All subjects in the experiments were native speaker of German, they were asked to rate the word pairs by similarity on a scale of 0-4.

- **ZG222** contains 222 word pairs along with their relatedness scores assigned on a discrete 0-4 scale by 21 subjects. The inter-annotator agreement is 0.49.

In our first experiment, we take the inverse of the NGD as the similarity score and compare them with the human score (shown in Figure 2-1,2-2,2-3).

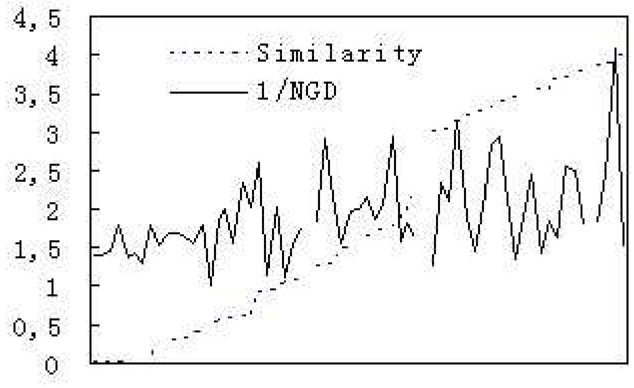


Figure 2-1: Human-rating vs. NGD (gur65)

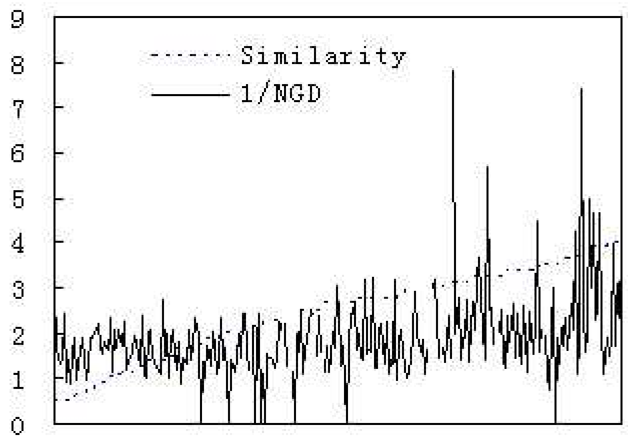


Figure 2-2: Human-rating vs. NGD (gur350)

We were somewhat discouraged by the poor correlation by the NGD-human agreement. We get two possible explanations for this:

1. The inverse of NGD might not be a good reflection of mapping distance to similarity. While we do have a convincing intuitive idea that the bigger the NGD is, the smaller the similarity is, we still lack a concrete function f to model this correlation. In addition to taking the inverse function, there are many more options, such as:

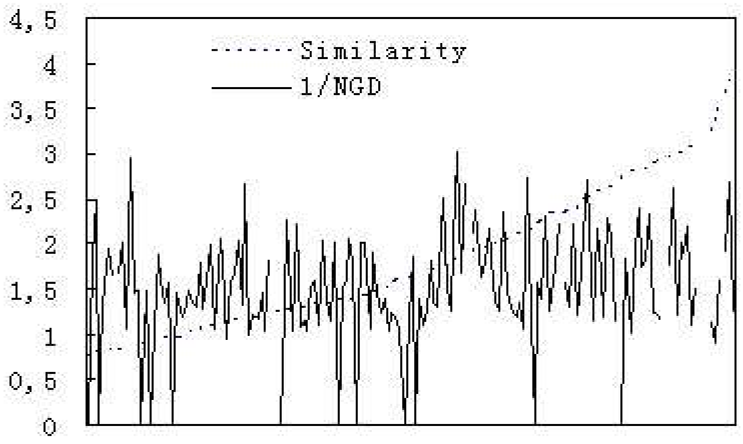


Figure 2-3: Human-rating vs. NGD (zg222)

- $f(x) = 1/x^2$
- $f(x) = -x$
- ...

It is hard to find the most precise model that reflects the exact relation between NGD and word similarity. The bad performance of taking the inverse does not necessarily mean that NGD is a bad indicator of word similarity.
In order to solve this problem, we decided to run some correlation tests on the original NGD data and the human rating to test the agreement on both sets, as a substitution of explicitly comparing the absolute value.

2. The NGD method did not take the inflected form of words, which are semantically the same as the original word, into consideration. (For the German dataset, we only queried for the base form (the lemma) not for all word forms due to German's complex morphology and the fact that we did not have a suitable morphological generator at hand.)

We conducted a second experiment on an English dataset[6], the WordSimilarity-353 Test Collection [7], querying for all combinations of inflected forms and using a correlation test to compare with the human ratings. We used two datasets:

- **Set1** contains 153 word pairs along with their similarity scores on a scale from 0 (totally unrelated words) to 10 (very much related or identical words) assigned by 13 subjects.
- **Set2** contains 200 word pairs with their similarity scores on a scale from 0 to 10 assigned by 16 subjects.

The morphology generation tool we used was RASP[8]. Nouns are inflected to *singular* and *plural*, while verbs are inflected to *present, 3rd person singular, present participle, past, past participle.* Queries are expanded to all combinations of inflected forms, as shown in Example 2.8.

(2.8) singer AND sing $\rightarrow$ singer AND sing, singer AND sings, singer AND singing, singer AND sang, singer AND sung, singers AND sing, singers AND sings, singers AND singing, singers AND sang, singers AND sung,

As mentioned above, we utilized a correlation test in order to avoid the problem of how to choose a explicit form for mapping NGD to similarity:

Spearman's rank correlation coefficient[27], named after Charles Spearman, is a non-parametric measure of correlation. It assesses how well an arbitrary monotonic function can describe the relationship between two variables, without making any assumptions about the frequency distribution of the variables.

In principle, Spearman's rank correlation coefficient test ranks the data before calculating the coefficient. The differences between the ranks of each observation on

[6]The English dataset was chosen due to the easy usability of the morphological generator from RASP [7].

[7]Available from the computer science department of *Technion - Israel Institute of Technology:* http://www.cs.technion.ac.il/ gabr/resources/data/wordsim353/

[8]It was originally developed on a UK EPSRC-funded project, and is available from http://www.informatics.susx.ac.uk/research/groups/nlp/rasp/

the two variables are calculated (as shown in fomula 2.9).

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{2.9}$$

where:

- d_i is the difference between each rank of corresponding values of x and y, and
- n is the number of pairs of values

In the ideal case, there is no difference in the rank of the corresponding values of the two sets, with correlation coefficient ρ being 1 in this case, which is evidence of a strong correlation between the two data sets.

Kendall tau rank correlation coefficient [20] is a non-parametric statistic used to measure the degree of correspondence between two rankings and assess the significance of this correspondence. In other words, it measures the strength of association of the cross tabulations. It is defined as:

$$\gamma = \frac{2P}{0.5n(n-1)} - 1 \tag{2.10}$$

where:

- n is the number of items, and
- P is the sum, over all the items ranked after the given item by both rankings

The Kendall tau coefficient falls into the range $(-1, 1)$, with bigger numbers indicating stronger agreement.

Not all the word pairs can be processed due to the fact that the search engine APIs cannot process high frequency terms properly (see Section 2.1.3). The recall rate for *Set1* is 63.6% (98 out of 154), while it is 49.3% for *Set2* (99 out of 201).

All the experimental result tables shown in the following part of this section are based on data *Set1*. We obtained quite similar results from *Set2*.

Some conclusions drawn from the experimental results running on Spearman's correlation test:

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
h1	1	0,73	0,75	0,69	0,7	0,62	0,71	0,78	0,78	0,8	0,63	0,72	0,68
h2		1	0,74	0,75	0,65	0,49	0,7	0,78	0,7	0,75	0,62	0,75	0,72
h3			1	0,82	0,65	0,71	0,76	0,79	0,75	0,74	0,65	0,73	0,78
h4				1	0,71	0,61	0,74	0,77	0,67	0,7	0,62	0,69	0,79
h5					1	0,6	0,72	0,66	0,7	0,67	0,62	0,7	0,73
h6						1	0,58	0,57	0,59	0,58	0,41	0,54	0,61
h7							1	0,74	0,71	0,72	0,56	0,78	0,78
h8								1	0,74	0,77	0,63	0,79	0,73
h9									1	0,77	0,58	0,76	0,82
h10										1	0,69	0,77	0,77
h11											1	0,66	0,65
h12												1	0,8
h13													1

Table 2.3: Human-human Spearman's correlation: the i^{th} subject (h_i)

- The inter-annotator agreement fluctuates a lot among different human ratings(see Table 2.3).
- The agreement between individual human and average human rating (AHR) is high, with most cases above 0.8(see Table 2.4).
- Although the agreement between individual humans and the NGD is slightly discouraging, with a minimum agreement of 0.19 and a maximum agreement value of 0.58, it still exceeds the worst human-human agreement. In a sense, it is comparable to human rating.
- The agreement between the NGD and the AHR is acceptable, considering the fact that the worst human-human agreement (0.41 between $h6$ and $h11$) is much lower than the NGD-AHR agreement of 0.58. In the experiment run on the second data set *Set2*, we got a NGD-AHR agreement of 0.52, with the worst inter-annotator agreement being 0.25.

It might be the case that different correlation tests may influence the result of the correlation coefficient. We also did the *Kendall tau test* to compare with the

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
AHR	0.85	0,84	0,86	0,84	0,8	0,67	0,84	0,88	0,85	0,9	0,75	0,87	0,89

Table 2.4: Human-AHR Spearman's correlation: the i^{th} subject (h_i), average human rating (AHR)

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
NGD	0.36	0,42	0,52	0,48	0,34	0,48	0,48	0,49	0,47	0,42	0,19	0,44	0,58

Table 2.5: Human-NGD Spearman's correlation: the i^{th} subject (h_i), normalized Google distance (NGD)

Spearman's ranking test (see result from Table 2.7, 2.8, 2.9, 2.10).

The result from the Kendall tau test further confirmed the result attained from the Spearman's test: although the human-AHR agreement is high, the human-human agreement fluctuates between the different individuals, with some agreement as low as 0.35. The human-NGD agreement is comparable to the inter-annotator agreement.

We also did an experiment to test the dependency vector method (see Section 2.1.2). The experiment used the tool provided by Pado and Lapata and Lapata [29, 30]. The training corpus that was used for the dependency vector method is the British National Corpus (BNC).

Table 2.11 and 2.12 shows the Spearman's correlation test result between human rating and the dependency vector based similarity, while Table 2.13 and 2.14 show the results of Kendall tau correlation test. The performance of the Google distance measure beats the performance of the dependency vector method in both of the tests.

	NGD
AHR	0.51

Table 2.6: AHR-NGD Spearman's correlation: normalized Google distance (NGD), average human rating (AHR)

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
h1	1	0,58	0,63	0,57	0,63	0,59	0,59	0,65	0,64	0,7	0,52	0,6	0,6
h2		1	0,66	0,68	0,62	0,5	0,62	0,71	0,6	0,68	0,53	0,67	0,64
h3			1	0,71	0,61	0,68	0,66	0,69	0,64	0,66	0,56	0,63	0,7
h4				1	0,65	0,59	0,64	0,67	0,57	0,63	0,56	0,59	0,72
h5					1	0,58	0,61	0,58	0,59	0,61	0,53	0,6	0,64
h6						1	0,49	0,5	0,49	0,53	0,35	0,45	0,53
h7							1	0,66	0,61	0,67	0,48	0,68	0,69
h8								1	0,64	0,7	0,54	0,69	0,64
h9									1	0,7	0,51	0,65	0,72
h10										1	0,58	0,66	0,67
h11											1	0,58	0,59
h12												1	0,72
h13													1

Table 2.7: Human-human Kendall tau correlation: the i^{th} subject (h_i)

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
AHR	0,84	0,73	0,79	0,77	0,76	0,66	0,75	0,81	0,76	0,84	0,68	0,8	0,82

Table 2.8: Human-AHR Kendall tau correlation: the i^{th} subject (h_i), average human rating (AHR)

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
NGD	0,2	0,28	0,34	0,31	0,2	0,29	0,31	0,31	0,29	0,26	0,1	0,29	0,38

Table 2.9: Human-NGD Kendall tau correlation: the i^{th} subject (h_i), normalized Google distance (NGD)

	NGD
AHR	0.34

Table 2.10: AHR-NGD Kendall tau correlation: normalized Google distance (NGD), average human rating (AHR)

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
DV	0,21	0,34	0,26	0,27	0,28	0,32	0,3	0,34	0,24	0,2	0,12	0,32	0,35

Table 2.11: Human-DV Spearman's correlation: the i^{th} subject (h_i), dependency vector (DV)

	DV
AHR	0.31

Table 2.12: AHR-DV Spearman's correlation: average human rating (AHR), dependency vector (DV)

Although the NGD has a problem with recall due to the fact that high frequency words cannot be dealt with the search engine API, it has good performance on the records that it can deal with.

We believe that the main reason that the web counts method outperforms the dependency vectors for computing semantic relatedness is that the web is a significantly larger database than any compiled corpus, which makes it more likely that we can find information about semantically related concepts, alleviating data sparseness. Furthermore, the web undergoes much more up-to-date changes than any linguistic corpus. It has also been shown that web counts can be used as reliable proxies for corpus-based counts and often lead to better statistical models [42, 21].

A second cause for worse performance of the dependency vector method is the problem of *word sense ambiguity.* Unlike the lexical knowledge-based method, the statistical-based approach does not do any word sense disambiguation (see Section

	h1	h2	h3	h4	h5	h6	h7	h8	h9	h10	h11	h12	h13
DV	0,11	0,22	0,17	0,17	0,16	0,19	0,2	0,22	0,15	0,12	0,07	0,21	0,23

Table 2.13: Human-DV Kendall tau correlation: the i^{th} subject (h_i), dependency vector (DV)

	DV
AHR	0.21

Table 2.14: AHR-DV Kendall tau correlation: dependency vector (DV), average human rating (AHR)

2.1.1). The assumption is that the words occurring in the statistical corpus share the same sense with the words in the application task. In reality, the statistical approaches conflate different word senses by summing over all senses with the weight of corresponding occurrence frequency in the corpus (modeled as Formula 2.11). A certain sense of the word w (general concept), $S(w_k)$, in a context is equal to the statistical semantic score if-and-only-if the occurrence of all other senses is equal to 0 (see Formula 2.12), which is often not the case.

$$S(w) = \sum_i f(w_i)S(w_i) \tag{2.11}$$

where, w_i is the i^{th} sense of the general concept w, $S(w)$ is the conflated sense of w, $f(w_i)$ is the frequency of the sense w_i.

$$S(w_k) = \sum_i f(w_i)S(w_i) \quad if.f.\ f(w_i) = 0, i \neq k \tag{2.12}$$

(2.13) I will go to the *bank* to withdraw some money.

(2.14) The other *bank* of the river is much more beautiful.

Example 2.13 shows *bank* interpreted as *a financial institution*, while 2.14 shows *bank* interpreted as *the slope immediately bordering a stream course along which the water normally runs.* If *bank* occurs twice, with the two semantic interpretations each, in the corpus, the output statistical semantic score would be combined senses with the two senses each weighting 0.5 (see Formula 2.15). This result does not correspond to any of one specific sense, which makes the effectiveness of our algorithm depend on sense distribution of the corpus. If we are lucky, most of the occurrence of the word in the corpus share the same sense with our application task, the conflated sense is a close approximation. However, a false approximation can happen due to a sense distribution bias in the statistical corpus.

$$S(bank) = 0.5S(bank_f) + 0.5S(bank_s) \tag{2.15}$$

We leave this problem open due to the complexity of word sense disambiguation in a large corpus, which is not the focus of our topic.

A distributional bias of different senses, such as one sense dominating over the others, can also lead to bad performance by the inconsistency of the dominating sense in the corpus and the sense in the application task. A relatively smaller corpus (comparing BNC with the whole web) might have more chance toward a sense distribution bias problem, while a large corpus such as the web, which has a much more balanced database, might have less risk of predicting the semantic score based on one specific sense. This might be another reason for the worse performance of the dependency vector method compared with NGD.

We decided to choose NGD, instead of lexical knowledge or dependency vector based approaches, to model semantic relatedness for all the work in the subsequent parts of this thesis, based on the experiment results described in this section.

2.2 Chaining Algorithm

In this section, we describe how to make use of the semantic relatedness measure discussed in Section 2.2 to build lexical chains for our MWEs literal or non-literal use identification task.

The chaining algorithm follows three steps in general:

1. Choose candidate words
2. Place new words into existing chains
3. Optimization based on the chain scores

2.2.1 Candidate Words

The basic principle to choose a candidate word is to choose word classes that are more informative, while filtering the words that are less informative.

Hirst and St-Onge [16] choose words which are nouns contained in WordNet, and do not appear in the stop-word list. The stop-word list contains closed-class words and many vague high-frequency words that tend to weaken chains by having little content. Morris and Hirst [26] use a similar strategy, eliminating closed-class words such as pronouns, prepositions and verbal auxiliaries, as well as high-frequency words. Jarmasz and Szpakowicz [18] only consider nouns, adjectives, verbs, adverbs and interjections which are in Roget's thesaurus. Brunn et al. [8] design a heuristic using the idea that nouns contained within subordinate clauses are less useful for topic detection than those contained within main clauses. They filter the nouns that appear in the subordinate clauses, and select the candidate words which come from an open class of words that function as a noun phrase or proper name after the noun filtering process. Some studies only extract content words, such as nouns, verbs, adjectives and adverbs [9, 28].

We included as many words as possible in our candidate chaining words, while some very high frequency words (typically function words) were excluded naturally due to the data overflow problem of the search engine API (see Section 2.1.3). This allows us to model the context as well as possible, while at the same time reducing the computational cost by disregarding high frequency words which contribute little to the overall semantics of the context.

2.2.2 Chain Forming

In the chain forming process, all candidate words of the text are considered in sequence and for each word it is determined whether it is similar enough to one of the existing chains to be placed in that chain, if not, it starts a new chain of its own.

The chain forming part needs to address two issues: choosing a good *relatedness threshold*, and *conflict resolution*.

The **relatedness threshold** defines the minimum semantic relatedness score that is necessary to place a word in a chain. In other words, a new word is placed into an existing chain only if the semantic relatedness between the word and the chain is above the threshold, otherwise this word starts a new chain.

The relatedness threshold can be modeled as *word-chain relatedness* or *word-word relatedness*. While the former takes the whole chain as a semantic-bearing unit, the latter is more focused on how the new word is related to individual nodes in the existing chain.

There are three different ways to define *word-chain* semantic relatedness:

- Sum: add up all the semantic relatedness scores between each word in the chain and the candidate word
- Average: normalize the sum value by dividing the number of relatedness pairs
- Maximum: take the maximum of all the relatedness pairs[9]

Although there are studies showing that the length and strength of the chain are positively correlated [4], method *sum* may put so much preference on longer chains, hence shorter chains may never get the chance to gain new members. The consequence is that longer chains may act as a snowball as the chaining process is going on, noisy data can be easily integrated into existing chains which leads to further more false insertion.

The *average* strategy aims to solve this problem by introducing a normalization factor. However, the semantically badly connected chain nodes (false members) in the *average* strategy may prevent some real chain members from added by this normalization factor, even though there are strong links between the current word and some existing chain members. In a sense, this method can have a slight preference on short chains.

The *maximum* method may be sensitive to noisy data. Some false chain members may easily lead to new false data being inserted due to a strong semantic link with

[9]The *maximum* method can also be categorized in the word-word relatedness strategy, because it only picks the relatedness of one single word from the chain instead of combining them together.

the existing false members. Our experimental results supported this argument.

An alternative method, *word-word relatedness*, is to look for the pair wise mutual semantic relatedness between the word and each chain word. The new word is placed into the chain only if it is related to *every word* in the chain, which is to say, every pair of the relatedness score should be above the threshold.

The *word-chain* method blurs the effect of individual nodes by mixing the effect of all words as a whole. We believe the *word-word* based relatedness threshold is more reliable, since it keeps a clear record of every pair wise mutual word-word relatedness. Any machine learning algorithm is optimized on more training instances based on the fact that the number of word-word samples is much greater than word-chain samples.

We implemented the *word-word relatedness* and the *maximum* strategy of the *word-chain relatedness* in our experiment, and found that the chains attained from the *maximum* strategy are long and intuitively unreliable, while the more conservative strategy, *word-word relatedness*, leads to shorter but more reliable chains (see sample chains in Appendix A).

Conflict resolution is needed when there exist multiple suitable chains for the candidate word to be placed into. One alternative is to place the new word in the chain that gains the maximum relatedness score (greedy search). The problem, then, is that all placement of new candidate words is only based on the previous observations, even though there might be strong evidence in the later context that a former inserted word actually belongs to another chain. An alternative strategy is to insert the new word in several chains where the overall relatedness gains are above the threshold. The chain building process keeps all the alternatives until the end, and then words are pruned from the chains where the deletion of them leads to less drop of the overall chain score. This strategy is better in the sense that it keeps an eye on the backward context, i.e., the context that follows . The fact that backward context can possibly renew the overall score of the chain reduces the risk of discarding the global optimum chain solution. The disadvantage is that, if too many potential chains are kept for a certain word, the lower ranking chains, in which the word does not really belong, will be largely influenced by noisy data, which is caused by the fact that an earlier false insertion may introduce more false insertions by the semantic relatedness with the false chain members.

Theoretically, the ideal case is a trade off between these two conflict resolution strategies can be found. It is a beam search[10] problem. The general principle is to keep a certain number of potential candidates in the chaining pool so that the backward context can play a role, avoiding the solely forward context dependent greedy search, while at the same time, limiting the number of potential candidates for reducing the risk of introducing noisy data.

We think it might be possible to achieve this goal by adjusting the relatedness threshold. While a big threshold gives more allowance to backward context, a smaller threshold puts more restrictions on insertions, reducing the risk of introducing more false chain members. The second strategy can be accomplished by limiting the number of candidate chains in the chaining pool (chaining pool threshold); for example, only the top-n ranking chains can be selected.

We did not go into much detail in testing how different relatedness threshold and chaining pool threshold values influence the performance of the chaining algorithm in our implementation (which is an interesting thing to do in the future). At the moment, we simplified this part by a greedy solely forward context dependent algorithm, always placing the word into the nearest chain, where the pair wise word-word semantic relatedness is above the threshold (see Section 2.3.3).

2.2.3 Chain Scoring

Whether the target expression is used literally or not is decided based on the computed chains described in Section 2.2.2. A simple strategy would be to classify an expression as literal whenever it appears in any existing chain. However, this might predict too many literal cases, because many of the component words in non-literal MWEs can participate in weak chains as well. Therefore, we need to design a chain scoring

[10]Beam search is a heuristic search algorithm that is an optimization of best-first search that reduces its memory requirement. In beam search, only a predetermined number of best partial solutions are kept as candidates [40].

strategy to evaluate the strength of the chains for the chain-based classifier. Literal usage is then only predict if an MWE participates in a chain whose strength is above a threshold.

Different approaches for scoring chains have been proposed:

Morris and Hirst [26] found three factors contributing to chain strength: *reiteration* (the more repetitions, the stronger the chain), *density* (defined as the number of chain members divided by the span of the chain members across the context. the denser the chain, the stronger it is), and *length* (the longer the chain, the stronger it is).

Barzilay and Elhadad [4] use *length* and *homogeneity* as predictors of the strength of a chain. They define *length* as the number of occurrences of members of the chain, and *homogeneity index* as the number of distinct occurrences (words) divided by the length. The score of the chain is defined as the product of the length and the homogeneity index.

Silber and McCoy [35] designed a scoring system that allows different types of relations within a lexical chain to contribute to that chain differently. They use a scoring system tuned by empirical methods.

Some chain scores are calculated based on the number of repetitions and the type of WordNet relations between chain members, which means that the chain score is the sum of each word pair in the chain [12]. Each word pair's score is calculated as the sum of the frequencies of the two words, multiplied by the relationship score between them.

Okumura and Honda [28] evaluate the strength of the chains by *recency* and *length.* More recently updated chains are considered to be the more activated context in the neighborhood and are given more salience. Longer chains are considered to be more about the topic in the neighborhood and are given more salience.

It might be interesting to test different strength scoring factors mentioned in this section in the future, but at the moment we evaluate the strength of the chain only based on the *length.* The **classification threshold** is defined as the minimum length requirement a chain has to be for predicting a literal use. We need annotated

MWEs	Nonliteral	Literal	All
bite off more than one can chew	142	2	144
back the wrong horse	25	0	25
blow one's own trumpet	9	0	9
bite one's tongue	150	16	166
*bounce off the wall	7	39	46
break the ice	521	20	541
*drop the ball	215	688	903
get one's feet wet	140	17	157
pass the buck	255	7	262
play with fire	532	34	566
*pull the trigger	4	11	15
rock the boat	470	8	478
set in stone	272	9	281
spill the beans	172	3	175
sweep under the carpet	9	0	9
swim against the tide	125	1	126
tear one's hair out	54	7	61
Total	3102	862	3964

Table 2.15: Experimental data (* indicates expressions for which the literal usage is more common than the non-literal one)

development data to get this threshold (see Section 2.3.3 for detail).

2.3 Experiment and Evaluation

2.3.1 Data

We used an existing data set in which occurrences of 17 potentially idiomatic expressions were labeled as literal or non-literal[11]. The context of an example consists of five paragraphs, with the current paragraph containing the expression plus two preceding and following ones (see 2.16).

(2.16) ⟨*idiom="bounce off the wall" file="apw200710" label="l"*⟩

[11]This data set was built as part of the SALSA project in Saarland University: http://www.coli.uni-saarland.de/projects/salsa/

After a slow start to the season, the Giants (34-32) are picking up speed with big contributions from their two slugging stars.
Bonds also doubled and drew two walks from Valdes while scoring three runs.
Bonds, who struck out three times against Scott Schoeneweis on Wednesday night, and Kent scored three runs apiece.
Kent, who had two hits, got a triple in the fourth when Darin Erstad badly misjudged Kent's drive to the warning track in center. Blinded by the sun, Erstad leaped at the ***wall***, but the ball ****bounced off the wall**** well below his glove.
J.T. Snow, who returned from the disabled list for his first game since May 26, capped San Francisco's five-run fourth with an RBI groundout.
With ample run support, Mark Gardner (2-5) allowed seven hits and two runs over five innings for just his second victory in 13 starts this season.

2.3.2 Baseline

The distribution of the data is unbalanced (78% out of 3964 examples are used idiomatically). The non-literal instances are almost three times as many as the literal ones, which means any method that has a preference of predicting non-literal use would have a strong advantage under evaluation. We built our first baseline, *B1*, based on this observation, always predicting non-literal use. Due to the large percentage of non-literal cases in the experiment data, the F-measure of the non-literal instances is high, which makes beating the performance on the non-literal case hard for any other method. A reasonable improvement strategy is to increase the performance on the literal cases while still maintaining a good one on the non-literal cases. We built a second baseline based on this priciple.

The intuition behind the second baseline, *B2*, is that literal repetition of MWE component words is a good indicator of literal use. Take the MWE *bounce off the wall* as an example, if the forward context mentions *wall* and *ball*, then the succeeding appearance of the MWE *bounce off the wall* is more likely to be of literal use, meaning that *the ball is bounced off the wall*. One of these examples is shown in Example 2.16.

Method	Lable	Prec.	Rec.	$F_{\beta=1}$
B1	N	0.78	1.00	0.88
	L	–	–	–
B2(N)	N	0.84	0.95	0.89
	L	0.69	0.40	0.50
B2(V)	N	0.83	0.95	0.88
	L	0.70	0.40	0.51
B2(N+V)	N	0.83	0.90	0.86
	L	0.58	0.43	0.50
$B2^*_p$(N)	N	0.79	0.98	0.87
	L	0.77	0.19	0.30
$B2^*_p$(V)	N	0.76	0.98	0.86
	L	0.36	0.04	0.08
$B2^*_s$(N)	N	0.77	0.99	0.86
	L	0.67	0.06	0.11

Table 2.16: Baseline (B1: always predict literal; B2(N): predict literal based on literal repetition of noun; $B2^*_p$(V): predict literal based on literal repetition of verb in the current paragraph)

In contrast, if the context does not contain any repetition of the MWE component words, the MWE is more likely to be an idiomatic usage (see 2.17).

(2.17) ....

> "I have an excellent relationship with Joel," she remarked at the Oscar nominees luncheon. "He tends to be the bedrock. I have the freedom to ***bounce off the wall*** if necessary, and he's there to clean up the mess."
> ...

(2.18) ...

> "He's our high-maintenance kid, ***bouncing off the wall***, he's got so much energy," said his father, Walt Weiss, a reserved and reflective man.
> "Everybody who knows him always jokes about it. The first thing people who haven't seen us for a while is, 'How's Brody?' Everybody loves him because he's so active."
> Walt Weiss is leaning against a ***wall*** down the hall from the Intensive Care

Unit at Scottish Rite Children's Medical Center, which is all wrong. The past week has been all wrong, not just for Walt and wife Terri Weiss, but for a small but heavy handful of families in the metro Atlanta. Their kids have been diagnosed with E. coli 0157:H7, a deadly strain of bacteria.

...

There are counter examples as well. It is possible that a literal repetition of an MWE component word appears in the non-literal cases. In Example 2.18, the literal repetition of *wall* in the backward context serves as a false evidence of literal use.

Many MWEs contain some functional words, such as *the, in, under.* Functional words have a high repetition rate due to their common use in English. We exclude them from *B2*, as they convey little semantic meaning of the MWE, and including them may lower the precision of literal prediction.

Intuitively, nouns and verbs are the two main semantic-bearing word categories in our experimental data. We focus on the literal repetition of these two types, and did different experiments on noun, verb and the combination of both to test how different part-of-speech affects the result (as shown in the third row of Table 2.16). While the performances of choosing nouns vs. verbs are close, the combination of both decreases the precision at a gain of recall.

We found that the distance between the literal repetition and the MWE may be an indicator of the true or false positive evidence of literal usage by comparing Examples 2.16 and 2.18. The closer the repetition is, the more likely it shares the same semantic meaning of the MWE component word (see 2.16). In the counter cases, the repetition may occur in a different paragraph as in 2.18.

We did another experiment to test it with all the data by introducing a window function. It is defined by Equation 2.19, which means that any token repetition t that occurs within a distance that is below the threshold T from the MWE is considered. All literal repetition outside the window is disregarded in *B2*.

$$W(t) = \begin{cases} 1 & \text{if } |t - MWE| < T \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

The default window function is all the context in the data examples. We tried with two other window functions, a paragraph function p and a sentence function s. Only literal repetition within the same paragraph is considered in the p function ($B2^*_p$), while repetition is only considered within the same sentence in the s function ($B2^*_s$).

It can be seen that different window functions mainly affect the precision and recall rate of the literal case. There are two reasons: firstly, the non-literal cases account for the major proportion of the data, so the performance is not as sensitive as the literal cases. Secondly, the recall rate for the literal cases is always low, which means that *B2* has a strong tendency to predict non-literal use no matter which window function is chosen (the truth is that literal repetition in not common in the experimental data).

(2.20) ..

> But others liken a coalition that includes the king and the Northern Alliance to the Soviet Union's attempts to prop up unpopular political bosses during the 1980s.
>
> "They ***backed the wrong horse***", Khan said of the Soviets, "and the Americans are backing a similar ***horse***. It will be very hard replacing the Taliban with the Northern Alliance or an imposed person like Zahir Shah."
>
> ...

Furthermore, the smaller the window is, the lower the recall rate for the literal use, but the precision rate is not positively correlated with the size of the window. The sentence window function gets a lower precision rate compared with the paragraph function.

Intuitively, the smaller the window is, the more likely that the literal repetition of MWE component words is evidence of the literal use. This means that the s window function should lead to a higher precision rate than the p function, but the experimental results do not agree with our intuition. One reason for this might be that some literal repetitions that occur close to the MWE are another idiomatic use of the same MWE (see 2.20). The literal repeat of the component word *horse* in the following context is actually part of another occurrence of the same MWE on a different inflected form, *backing a similar horse*. It can also be the case that the precision variance might be rather unreliable given the fact that there are a few literal predictions with a low recall rate.

One strategy to eliminate this kind of error is to discard literal repetition that occurs inside another occurrence of the same MWE, but, unfortunately, the annotated experimental data do not supply such information. Multiple occurrences of the same MWE in one context were treated as different examples. Each example only contains the MWE tag for itself. It is possible to use regular expressions to automatically recognize whether the repetition is inside another occurrence of the MWE, but we did not go into further detail as this is not the main focus of our work.

While Example 2.18 gives us a convincing example of choosing small windows over big ones for the precision of predicating literal usage, the multiple occurrence problem described in 2.20 gives us another proof of why a smaller window function does not necessarily lead to better precision rates at the cost of low recall. From this perspective, we can only choose a window function that is some kind of compromise in the middle, with acceptable precision and recall rate. We believe the *paragraph* function is the best choice.

The problem caused by whole MWE repetition leads to that the result of the window function is not as ideal as we thought. Even at the cost of a very low recall rate, the precision for predicting the literal use is still be lower than a larger window function. The main reason is that whole MWE repetition usually occurs inside a small window, and they contribute to the false positive counts in any window function. The percentage of the whole MWE repetition instances determines how the precision rate curve responds to the increasing or decreasing of the recall rate.

Among the semantic-bearing word categories verbs are especially interesting. Table 2.16 shows that the performance of verb in $B2$ varies a lot while the window function is tuned. The complete context window function gets a precision of 0.70, while the paragraph window function gets a precision of 0.36. We choose noun over verb as the semantic-bearing word category for detecting literal repetitions based on the observed instability of the performance of verb.

Although $B2(N)$ gets the best overall performance of all three noun-related methods, we choose the $B2^*_p(N)$ method as the final version of Baseline 2. The reason is that we want Baseline 2 to be a method that has a high precision in predicting literal use even if this is at the cost of low recall, and we want to combine with some other methods which take the output of Baseline 2 as input in the future. We think the precision of the literal usage case by Baseline 2 is hard to beat due to the reason that close distance literal repetition of the semantic-bearing MWE component words is strong evidence of literal use (whole MWE repetition is an exception). We seek to find a method that can improve the recall rate of the literal case, while maintaining a relatively high precision. In the next section, we describe the experimental result of the chain-based classifier.

2.3.3 Experimental Results

Parameter Optimization: the lexical chain method has two parameters, relatedness threshold (see Section 2.2.2) and classification threshold (see Section 2.2.3), that need to be optimized. We use a portion of the annotated data as a development set[12] (such as *break the ice*) for optimizing these parameters. The parameters were optimized on *accuracy* in *accuracy optimization*, *literal F-score* for *literal F-score optimization.* We did a exhaustive search to find the best combination of the parameters in the development set by setting the relatedness threshold in steps of 0.02, and the classification threshold in steps of 1. We chose the combination of thresholds that led to the best performance on the development set, and applied the optimized thresholds to the rest of the data.

In order to test how different development sets influence the parameter setting, and overall performance, we experimented with six different development sets:

- S1: *break the ice*, *bounce off the wall*

[12]Development set refers to the training set that we run the experiment on to find the best combination of classifier parameters.

	Development		Evaluation	
	Num.	LPerc.	Num.	LPerc.
S1	541	0.11	3377	0.24
S2	46	0.85	3918	0.21
S3	541	0.04	3423	0.25
S4	903	0.76	3061	0.06
S5	566	0.06	3398	0.24
S6	1107	0.05	2857	0.28

Table 2.17: Experimental data setup: different split of the development and evaluation set (S_i), number of instances (Num.), percentage of literal cases (LPerc.)

- S2: *bounce off the wall*
- S3: *break the ice*
- S4: *drop the ball*
- S5: *play with fire*
- S6: *play with fire*, *break the ice*

The number and the percentage of the literal cases of the development set and evaluation set are listed in Table 2.17. As seen from the data, most development sets had different distributional properties (the percentage of literal cases vs. non-literal cases) than the evaluation sets. The set that chose the combination of *break the ice* and *bounce off the wall* (see *S1*) as the development set leads to the most similar distribution between the development and evaluation set. Theoretically *S1* should get the best results, and our results supported this (see 2.19).

Table 2.18 shows the best results that are possible to gain on the development set by our specific chaining algorithm, which is based on accuracy optimization. For most data sets, we obtained over 0.9 accuracy on the development set. We further found that the performance on the development set tends to be lower when the literal percentage of the development data is high (see $S2$ and $S4$, both sets have higher literal percentage than other development sets). We think one possible explanation is

	Lable	Prec.	Rec.	$F_{\beta=1}$	Acc.
S1	N	0.93	0.99	0.96	0.92
	L	0.76	0.32	0.45	
S2	N	1.00	0.29	0.44	0.89
	L	0.89	1.00	0.94	
S3	N	0.97	0.99	0.98	0.97
	L	0.75	0.30	0.43	
S4	N	0.60	0.24	0.34	0.78
	L	0.80	0.95	0.87	
S5	N	0.97	0.99	0.98	0.96
	L	0.81	0.5	0.62	
S6	N	0.97	0.99	0.98	0.97
	L	0.77	0.44	0.56	

Table 2.18: Performance on the development set, parameters optimized on accuracy, precision (Prec.), recall (Rec.), F-Score ($F_{\beta=1}$), accuracy (Acc.)

that there is a certain amount of non-literal examples, where the MWEs component words participate strong lexical chains (see Example 2.18 and 2.20). When the number of literal examples increases, it is more difficult to find a certain combination of the *semantic relatedness* and *classification threshold*. The strategy has to be conservative enough so that not to predict the non-literal as literal where an MWE component word does participate in surrounding chains, while at the same time, it also has also to be greedy on literal predication so that the majority class (literal class) instances can be found.

Table 2.19 shows the results for the evaluation set with the thresholds tuned on the development data. As expected, in general the performance on the evaluation set is much worse than on the development set. The performance $S1$ gets the best results on the evaluation set, as the development and evaluation data are the most closely distributed. $S2$ got the worst performance of all, since it contains too many literal instances in the development set, and the percentage of literal cases varies most from the evaluation set. $S2$ has a strong tendency to predict literal usage, as a result of the literal case being the dominant class in the development, with a recall rate as

	Lable	Prec.	Rec.	$F_{\beta=1}$	Acc.
S1	N	0.85	0.89	0.87	0.80
	L	0.60	0.51	0.55	
S2	N	1.00	0.15	0.26	0.33
	L	0.24	1.00	0.38	
S3	N	0.81	0.92	0.86	0.79
	L	0.63	0.38	0.47	
S4	N	0.99	0.60	0.75	0.62
	L	0.12	0.86	0.20	
S5	N	0.78	0.96	0.86	0.77
	L	0.60	0.17	0.27	
S6	N	0.74	0.96	0.84	0.73
	L	0.59	0.17	0.26	

Table 2.19: Performance on the evaluation set, parameters optimized on accuracy, precision (Prec.), recall (Rec.), F-Score ($F_{\beta=1}$), accuracy (Acc.)

high as 1 but precision as low as 0.24.

We also did some optimization based on F-score on the literal class. Table 2.20 and 2.21 gives details. We found that the selected parameters deviate only minimally from those selected when optimizing for accuracy.

The optimized *relatedness thresholds* and *classification thresholds* of different optimization strategies and development sets can be seen in Table 2.22.

In order to test the upper bound performance of our chain-based classifier, we ran the development process on the whole data set. The best accuracy we can get is 0.84 (see Table 2.23 for detail). This classifier had a very restricted relatedness threshold and a comparatively loose classification threshold. We noted a similar performance by optimizing literal F-score, with a slight drop of non-literal performance and a small increase on the literal performance.

Both the *development-evaluation classifier* (split the data into two parts: development set for parameter optimization and test set for evaluation) and the *solely development classifier* (use the whole data as development set as well as evaluation set) outperforms the baselines.

	LPrec.	LRec.	$LF_{\beta=1}$	Acc.
S1	0.58	0.58	0.58	0.91
S2	0.89	1.00	0.94	0.89
S3	0.70	0.35	0.47	0.97
S4	0.78	0.98	0.87	0.78
S5	0.81	0.50	0.62	0.96
S6	0.77	0.44	0.56	0.97

Table 2.20: Performance on the development set, parameters optimized on the F-Score of literal class, precision (Prec.), recall (Rec.), F-Score ($F_{\beta=1}$), accuracy (Acc.)

	LPrec.	LRec.	$LF_{\beta=1}$	Acc.
S1	0.48	0.77	0.59	0.75
S2	0.24	1.00	0.38	0.33
S3	0.62	0.26	0.37	0.78
S4	0.10	0.91	0.18	0.51
S5	0.60	0.17	0.27	0.77
S6	0.59	0.17	0.26	0.73

Table 2.21: Performance on the evaluation set, parameters optimized on the F-Score of the literal class, precision (Prec.), recall (Rec.), F-Score ($F_{\beta=1}$), accuracy (Acc.)

One interesting observation from using the whole data set as development data is that while delivering similar results, the accuracy optimization got a strict *relatedness threshold* and loose *classification threshold*, but the F-score optimization on the other hand, got a loose *relatedness threshold* with a very conservative literal prediction threshold (*classification threshold*). In theory, it is possible to get the same performance by setting different combinations of the two thresholds. Thresholds give a lot room to play around with in our chain-classifier; a detailed study of how to make different combinations and how to choose a specific search strategy would be an interesting topic for future research[13].

[13]But all these work is on the assumption that we do have a reliable data source to evaluate automatically built chains.

	Accuracy		F-score	
	t1	t2	t1	t2
S1	0.64	4	0.72	5
S2	0.76	2	0.76	2
S3	0.62	4	0.6	4
S4	0.78	5	0.74	3
S5	0.58	4	0.58	4
S6	0.58	4	0.58	4

Table 2.22: Parameter settings: relatedness threshold (t1), classification threshold (t2), optimized on accuracy (Accuracy), optimized on the F-Score of literal class (F-score)

	N_{output}	L_{output}	$Prec.$	$Rec.$	$F_{\beta=1}$
N_{label}	2888	214	0.87	0.93	0.90
L_{label}	433	429	0.67	0.50	0.57

Table 2.23: Upper bound performance of the chain approach, parameters optimized on the whole dataset (accuracy)

	N_{output}	L_{output}	$Prec.$	$Rec.$	$F_{\beta=1}$
N_{label}	2593	509	0.91	0.84	0.87
L_{label}	267	595	0.53	0.69	0.61

Table 2.24: Upper bound performance of the chain approach, parameters optimized on the whole dataset (literal F-Score)

Chapter 3

Cohesion Graph-based Approach

The lexical chain based idea introduced in Chapter 2 serves to identify how the individual tokens of the MWEs participate in the context chains. There are three main problems with the lexical chain based approach:

- The choice of chaining algorithm for properly modeling the forward and the backward context. As discussed in Section 2.2.2, a trade-off strategy which compromises between forward and backward context needs to play around with different combinations of chaining thresholds, which is computationally expensive and cannot guarantee of a globally optimal solution. At the moment, we have only implemented a greedy search algorithm, taking the most likely solution based on the current context.

- The *relatedness threshold* and *classification threshold* co-effect the results of the chain-based approach, thus the performance is influenced not only by the quality of the chain but also the evaluation standard (*classification threshold*). We lack more explicit information about the effectiveness of the cohesion approach, as a poor performance may be caused by our chain scoring standard.

- Since we only have a small data set to apply the algorithm to, the performance of the data is sensitive to how the development set and evaluation set is split.

In this chapter, we introduce a cohesion graph to model the connectivity of the context. The chain approach is replaced by computing how the component tokens contribute to the overall connectivity of the cohesion graph. The idea is that if the component tokens of the MWEs decrease the connectivity of the context, the components may be assumed to be not closely semantically related to other words in the context. The MWEs should be considered as non-literal usage. For instance, the content word *ice* in contributes to the overall semantic connectivity of the whole sentence due to the fact that *ice* is semantically close to *water.* In contrast, the word *ice* in 3.2 decreases the overall connectivity as it is poorly connected to all the other words in this specific context (*play, party, games*). As a result, Example 3.1 is thought to be a literal usage, while 3.2 is thought to be an idiomatic usage.

(3.1) The water would *break the ice* surface into floes with its accumulated energy.

(3.2) We played a couple of party games to *break the ice.*

In the following part of this chapter, we give a formal definition of the cohesion graph that is used for identifying MWE as literal or non-literal.

3.1 Cohesion Graph Structure

The cohesion graph (CG) is an undirected complete graph [1] $G = (V, E)$, where

V: is a set of nodes $\{v_1, v_2, ..., v_n\}$, where each node $v_i = (w_i, f_i)$ represents a unique word in the context of the candidate MWEs. w_i is the surface form of the word, and f_i is the frequency of the word.

$$f_i = \frac{n_i}{N} \tag{3.3}$$

n_i is the number of occurrences of w_i, $N = \sum_{i=1}^{n} n_i$ is the total number of tokens in the context.

E: is a set of edges $\{e_{12}, e_{13}, ..., e_{(n)(n-1)}\}$, such that each edge e_{ij} connects a pair of nodes (v_i, v_j). The value of e_{ij} represents the semantic relatedness of the two words

[1] In the mathematical field of graph theory, a complete graph is a simple graph in which every pair of distinct vertices is connected by an edge. The complete graph on n vertices has n vertices and $n(n-1)/2$ edges.

w_i, w_j that e_{ij} connects.

$$e_{ij} = h(w_i, w_j) \tag{3.4}$$

h is a semantic relatedness assignment function. The explicit form of h will be given in the next section.

$c(G)$: is defined as the connectivity of the graph,

$$c(G) = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} (\lambda_{ij} \times e_{ij}) \tag{3.5}$$

λ_{ij} is the weight of e_{ij}.

The edges that connect more co-occurring words should be assigned higher weights according to the frequency. λ_{ij} is defined as

$$\lambda_{ij} = \delta(f_i, f_j) \tag{3.6}$$

where δ is a function that is correlated to f_i and f_j, with the constraint

$$\sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} \lambda_{ij} = 1 \tag{3.7}$$

Now the connectivity of the graph can be rewritten as:

$$c(G) = \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} [\delta(f_i, f_j) \times h(w_i, w_j)] \tag{3.8}$$

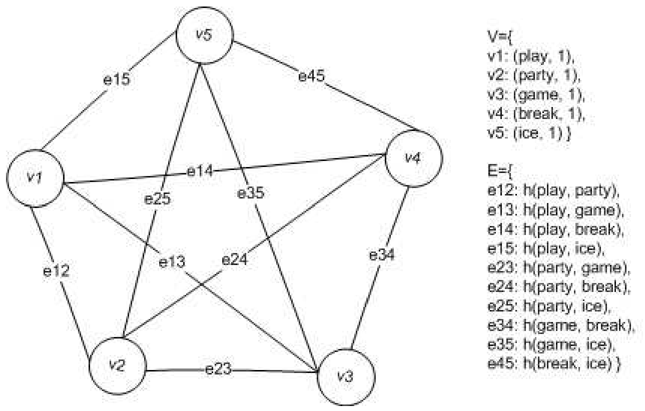
The algorithm for building a cohesion graph is described in Algorithm 1. Figure 3-1 gives an example of how a cohesion graph works.

3.2 Graph Connectivity

In Section 3.1, an abstract connectivity function of the cohesion graph has been given with the weight function δ and the semantic relatedness function h being left underspecified. Equation 3.8 says that the weight is influence by the frequency of

Algorithm 1: Cohesion graph building

```
input  : A context with a string of $n$ tokens $C = \{t_1, t_2, ..., t_n\}$
output : A cohesion graph $G = \{V, E\}$
1  Initialized the cohesion graph with $V_0 = \{\emptyset\}$, $E_0 = \{\emptyset\}$, and the dictionary
   $W_0 = \{\emptyset\}$;
2  for $i \leftarrow 1$ to $n$ do
3      if SemanticBearing($t_i$) then
4          if $\neg$FindWord($t_i$,$W_m$) then
5              Update $W_{m+1} = W_m \bigcup w_{m+1}$, where $w_{m+1} = t_i$;
6              $v_{m+1} = (w_{m+1}, 1)$, update $V_{m+1} = V_m \bigcup v_{m+1}$;
7              for $j \leftarrow 1$ to $|W_m|$ do
8                  Add $e_{(m+1)(j)} = h(w_{m+1}, w_j)$ to E;
9              else
10             $w_t \leftarrow$ FindWord($t_i$,$W_m$);
11             Update the frequency of the node $v_t = (w_t, f_t + 1)$;
```



V={
v1: (play, 1),
v2: (party, 1),
v3: (game, 1),
v4: (break, 1),
v5: (ice, 1) }

E={
e12: h(play, party),
e13: h(play, game),
e14: h(play, break),
e15: h(play, ice),
e23: h(party, game),
e24: h(party, break),
e25: h(party, ice),
e34: h(game, break),
e35: h(game, ice),
e45: h(break, ice) }

Figure 3-1: Cohesion graph for: *we play a couple of party games to break the ice*

the connected words (δ is a function of f_i and f_j) and the semantic relatedness is influenced by the surface form of the original words (h is a function of w_i and w_j). The information contained in the definition of the graph connectivity only reveals the factors that influence the overall connectivity. We need to investigate a more precise mathematic model to represent the graph connectivity.

In this section, the specific forms of the semantic relatedness function $h(w_i, w_j)$ and the weight function $\delta(f_i, f_j)$ are discussed.

As discussed in Section 2.1, there are many different ways to model semantic relatedness. In this task, we first tried with the *pointwise mutual information* (PMI) to avoid the total indexed number M problem in NGD (see Section 2.1.3), but the results indicated that PMI is much worse than NGD (see Section 3.5).

$$h(w_i, w_j) = \log \frac{P(w_i, w_j)}{P(w_i).P(w_j)} \tag{3.9}$$

The weight function $\delta(f_i, f_j)$ is a measure to assign a reasonable weight to the corresponding edge e_{ij}. We expect that higher frequency words are assigned higher weights so that they can contribute more to the connectivity of the cohesion graph. Based on this idea, one possibility to define δ is:

$$\lambda_{ij} = \frac{1}{Z}(f_i \times f_j) \tag{3.10}$$

where Z is a normalization factor. In this definition the weight w_{ij} is proportional to the product of the frequency of the two words t_i, t_j that the edge e_{ij} connects.

By the constraint defined in Equation 3.7, Z in Equation 3.10 can be written as:

$$Z = 1 - \sum_{i=1}^{n} f_i^2 \tag{3.11}$$

The connectivity of the cohesion graph in Equation 3.8 can be written as:

$$c(G) = \frac{1}{(1 - \sum_{i=1}^{n} f_i^2)} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} [f_i \times f_j \times h(w_i, w_j)] \tag{3.12}$$

An alternative strategy for defining the weight function $\delta(w_i, w_j)$ is to take the sum of the two frequencies. This definition gives less weight to edges connecting high frequency nodes. The weight is proportional to the sum of the two token frequencies instead of the product.

$$w_{ij} = \frac{1}{Z}(f_i + f_j) \tag{3.13}$$

By the constraint defined in Equation 3.7, the normalization factor can be calculated as:

$$Z = 2n - 1 \tag{3.14}$$

By this definition, the normalization factor is only relevant to number of unique words contained in the context. It is independent of the frequency of the words.

The connectivity of the cohesion graph in Equation 3.8 can be written as:

$$c(G) = \frac{1}{2(n-1)} \sum_{i=1}^{n} \sum_{j=1, j \neq i}^{n} [(f_i + f_j) \times h(w_i, w_j)] \tag{3.15}$$

Given an input text, the corresponding cohesion graph can be built with the graphing algorithm described in Section 3.1, and the overall connectivity can be calculated by Equation 3.12 or 3.15.

In our implementation, we utilized the 3.15 definition with the explicit form of $h(w_i, w_j)$ being PMI and NGD. It would be interesting to see how different connectivity functions influence the results, but we have not investigated this in great detail at the moment.

3.3 Graph-based Classifier

The idea of using cohesion graph for identifying MWE literal or non-literal use is to check how the MWE component words contribute to the overall connectivity of the context. In the literal use case, we supposed that each component word is well connected to other words, while in the non-literal use case, the component words are thought to be independent from the original text and are expected to have a low connectivity.

As shown in Figure 3-2(a), the edges that connect the MWE component words with other semantic-bearing words are labeled as key edges. The cohesion graph based MWE identification algorithm determines the contribution of these key edges to the overall connectivity.

A complete graph is built from the context tokens that surround the MWE (as shown in Figure 3-2(b)). New nodes from the MWE component tokens are added to the graph (nodes v_4 and v_5 in Figure 3-2(c)), and all the edges that connect the MWE component words and the surrounding context words are added to G ($e_{14}, e_{24}, e_{3,4}, e_{15}, e_{25}, e_{35}$ in Figure 3-2(c)).

Since the connectivity between the MWE component words does not contain any information of how these component words are semantically involved in the context, the updated cohesion graph is not a complete graph any more. All the edges between the MWE component words are deleted (as e_{45} in Figure 3-2(c)). We call the incomplete graph which contains a MWE as an *MWE Graph* (MWEG). The connectivity Formula defined for the complete graph in Equation 3.8 still holds, but the corresponding weight function for connecting between MWE component words, $\delta(f_i^{key}, f_j^{key})$ is set to be 0. The classifier of differentiating literal from idiomatic use is to check the connectivity gain Δc of the cohesion graph.

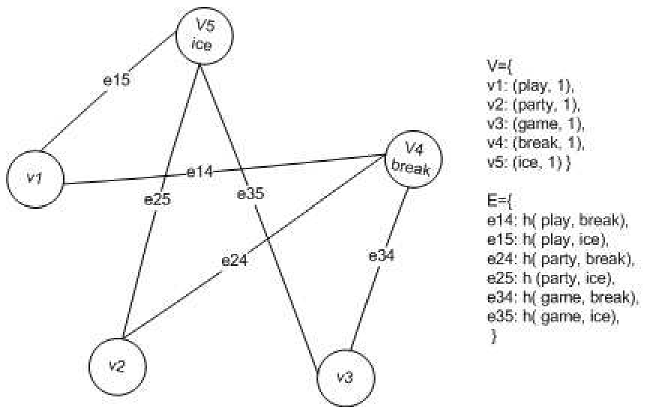
$$\Delta c = c(G) - c(G') \tag{3.16}$$

The formal process is described in Algorithm 2.

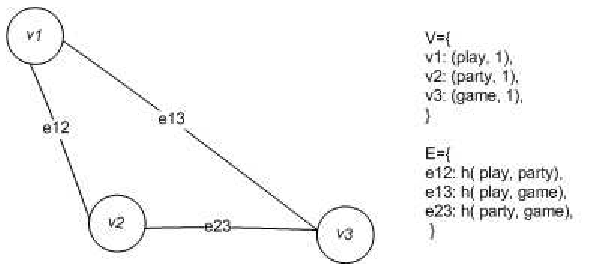
3.4 Pruning the Graph

There are two main problems with the cohesion graph: time complexity and noisy data, which can be further divided to four subproblems:

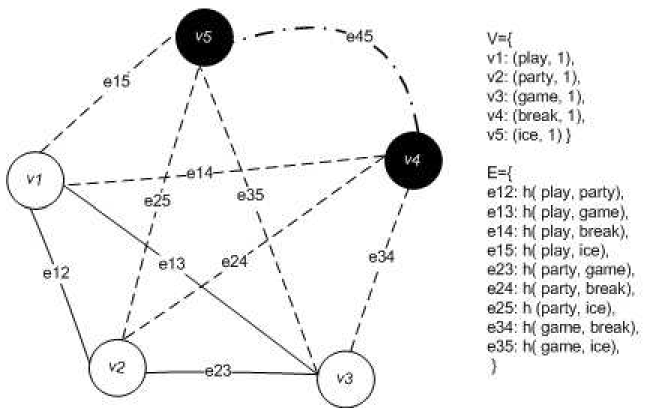
- The time complexity is $O(n^2)$.
- Functional words hardly bear any semantic meaning, but a corpus based statis-



(a) Key edges



(b) Cohesion graph without MWE



(c) Updated cohesion graph with MWE

Figure 3-2: MWE identification

Algorithm 2: MWE idiomatic usage identification

```
input  : A context $C$ with two sets of tokens, $K = \{k_1, k_2, ..., k_m\}$ is the set
         of component tokens in $MWE$, and $C' = \{t_1, t_2, ..., t_n\}$ is the
         context words without MWE. $C = K \bigcup C'$,
output: Isliteral($K$)
1  With the procedure described in Algorithm 1, build the cohesion graph
   $G' = \{V, E\}$ with the token set $C'$. $V = \{(w_i, f_i)\}_i$, and $E = \{e_{ij}\}_{ij}$;
2  Calculate the connectivity of $G'$ using Equation 3.12, $c(G')$;
3  for $i \leftarrow 1$ to $m$ do
4      Add new nodes $v_{|V|+i} = (k_i, 1) \rightarrow V$;
5      for $j \leftarrow 1$ to $|V|$ do
6          Calculate the semantic relatedness $h(k_i, w_j)$, and add edge
           $e_{(|V|+i)(j)} = h(k_i, w_j)$ to $E$;
7  Calculate the connectivity of the updated graph $G$, $c(G)$;
8  Calculate $\Delta c$ of the graph based on Equation 3.16;
9  if $\Delta c > 0$ then
10     Isliteral($K$) $=$ *true*;
11     else Isliteral($K$) $=$ *false*;
```

tical approach might assign a high semantic relatedness score $P(w, w_f)$ between the function word and other words. Luckily, the Yahoo search engine API helps us to filter out most of the functional words by returning the same 10-digit number (see Section 2.1.3).

- The weakly connected nodes or edges may indicate little semantic relatedness information in the context, but they can still influence the connectivity gain due to the normalization factor Z introduced in the weighting schema.

- Highly connected nodes might put too strong a bias on the overall connectivity, and decrease the effect of the MWE. As a result, connectivity gain would always be negative (Section 3.4.3).

The general principle of the pruning strategy is to get rid of the noisy data so that the connectivity gain is not influenced by irrelevant information. Based on the graph building algorithm described in Algorithm 1, there are two ways to prune the graph:

- Delete irrelevant nodes v_i.

- Delete irrelevant edges e_{ij}.

Sections 3.4.1, 3.4.2 and 3.4.3 discuss three special types of nodes that are noisy in our task, and how to prune them theoretically; in practice, we have done much on this part except some initial experiments such as deleting the top-3 weakly connected nodes (see Table 3.5). Detailed analysis of how to prune irrelevant edges has not been developed either. Pruning will be a main focus in our future work.

3.4.1 Semantic Bearing Nodes

Including words that are not semantically informative in the cohesion graph would influence the overall connectivity of the graph. Our general idea of choosing the semantic bearing words is to keep words that are more informative, while filtering out words that are less informative (the same as choosing candidate words in lexical-chain-based approach, see Section 2.2.1). We chose the semantic bearing words by leaving out all the high frequency words returned by Yahoo search engine API.

3.4.2 Weak Cohesion Nodes and Edges

Nodes that are poorly connected in the cohesion graph are defined as *weak cohesion nodes*. Edges that are poorly connected in the cohesion graph are defined as *weak cohesion edges*. As discussed above, weak cohesion nodes or edges can influence the overall connectivity of the graph due to the normalization factor Z introduced in the connectivity definition of the cohesion graph. A general pruning strategy for weak cohesion nodes is to eliminate them by discarding those whose overall connectivity to all the rest nodes in the graph is below a threshold. A similar threshold method can be applied to prune weak cohesion edges.

One disadvantage of this pruning strategy is that it does not change the time complexity of the graph building algorithm, $O(n^2)$. Since we do not have any prior knowledge of how a specific node is connected to the rest of the nodes in a context, the pruning strategy is done posteriorly (after the graph building). That means all the semantic relatedness values between all individual words have to be calculated in advance so that they can be used as a guideline for pruning.

One possible solution to solve this problem and lower the time complexity is to have a prior joint probability distribution P of the dictionary set W based on a linguistic knowledge base or statistical corpus.

If we suppose all the data in our task follows the same distribution as the prior distribution $P(W, W)$, then the top N ranking semantically related words W_{top} of a specific word w_k can be determined by Algorithm 3.

Algorithm 3: Determining top N semantic related words

```
1 Initialize $W_1 = W \;\; W_{top} = \{\emptyset\}$;
2 for $k \leftarrow 1$ to $N$ do
3     The $k^{th}$ ranking word $w_k$ is calculated by $w_k = \underbrace{max}_{j \in W_{k-1}} \{h(w_k, w_j)\}$;
4     Update $W_k$ with $W_k = W_{k-1} - w_k$;
5     Add $w_k$ to set $W_{top}$;
6 Return $W_{top}$;
```

A coarse-grained plan for pruning based on the prior probability distribution can be described as in Formulas 3.17 and 3.18. The first formula gives the principle for pruning weak nodes, and the second formula gives the principle for pruning weak edges. The idea is that if the overall semantic relatedness of a specific word with its top N related words is below a threshold, then this specific word is thought to be a weak semantic-bearing word, and should be excluded from the cohesion graph. Similarly, all the edges that satisfy Formula 3.18 are defined as weak edges, and should not be considered in the CG.

$$\sum_{i}^{N} P(w, w_i) < T, \qquad w_i \in W_{top} \tag{3.17}$$

$$P(w, w_t) < T, \qquad w_t \in W \tag{3.18}$$

Unfortunately, we do not have a prior distribution to refer to at the moment, so all the pruning is done posteriorly based on the experimental data itself. Detailed study of how to prune the graph would also be interesting to work on in the future.

3.4.3 Strong Cohesion Nodes

Definition 3.4.1. *A Strong cohesion word (SCW) is a word that has a strong semantic relatedness to all other context words.*

Suppose w_s is a strong cohesion word, $W = \{w_1, w_2, ..., w_n\}$ is the set of all context words, then Equation 3.19 holds:

$$h(w_s, w_i) >> h(w_j, w_i) \quad 1 \leq i \leq n; 1 \leq j \leq n; i \neq j \tag{3.19}$$

Definition 3.4.2. *Strong edges (SE) are edges that connects at least one node which is a strong cohesion word.*

The problem with strong cohension words is that they excessively contribute to the connectivity of the cohesion graph, eliminating or decreasing the effect of all the other words, including MWE component words. As a result, the strong cohesion words make our method which depends on graph connectivity gain caused by MWE component words less effective.

In the following part of this section, we first give the mathematical proof of the necessity of pruning SCWs, and then we give some possible solutions of how to prune them.

Theorem 3.4.1. *If $G^{'} = \{V, E\}$ is a complete graph, $v_s \in V$ is a strong cohesion word node, G is the corresponding MWEG of $G^{'}$, then the connectivity gain $\Delta c < 0$.*

Proof of Theorem 3.4.1.
Suppose:

- $G = \{V, E\}$ is the complete graph without any strong cohesion words and MWEs. $V = \{v_1, v_2, ...v_n\}$, $E = \{e_{12}, e_{13}, ..., e_{ij}, ...\}$;

- The average semantic relatedness between normal words w is ϵ;
- $v_s = (w_s, f_s)$ is the strong cohesion word node, and its semantic relatedness score with any other word is β, with $\beta >> h(w_i, w_j),\ w_i \neq w_s; w_j \neq w_s$;
- $v_m = (w_m, f_m)$ is the MWE component word;
- $G_1 \leftarrow add\ v_s\ to\ G$;
- $G_2 \leftarrow add\ v_m\ to\ G_1$.

Then:

- The connectivity of $c(G_1) = \frac{(n^2-n)\times\epsilon+2n\beta}{n^2+n}$;
- The connectivity of $c(G_2) = \frac{(n^2+n).\epsilon+2(n+1)\beta}{n^2+3n+2}$;
- The connectivity gain $\Delta c = c(G_2) - c(G_1) = B \times (\epsilon - \beta)$, where $B = \frac{2n(n+1)}{(n^2+3n+2)\times(n^2+n)}$;
- According to Formula 3.19, $\beta >> \epsilon$, so $\Delta c < 0$.

$\square$

Figure 3-3 gives a more specific example of how strong cohesion words influence the connectivity gain. v_5 is a strong cohesion word, and v_4 is a MWE component word. Intuitively, the proportion of strong edges out of all edges are constantly decreasing as the number of the nodes increases in the graph. In Figure 3-3, there are three strong edges, e_{15}, e_{25}, e_{35}, in graph $G_1 = \{v_1, v_2, v_3, v_5\}$. The proportion of strong edges is 3 out of 6. While in the MWE graph $G_2 = \{v_1, v_2, v_3, v_4, v_5\}$, the proportion of strong edges is 4/9. When adding new nodes, the number of total edges grows much faster than the number of strong edges.

We have not developed a practical strategy for pruning SCWs due to the fact that we need the information of all the semantic relatedness scores of each word pairs in the data instance to decide whether the definition in 3.19 holds. This requirement makes the practical process of this pruning strategy computational expensive. We have to try different thresholds to decide whether the overall connectivity of a specific node

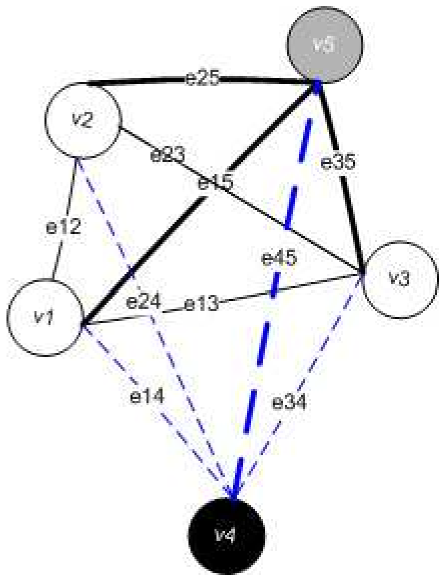


Figure 3-3: Example of strong cohesion word

to the rest nodes is strong enough. We can do this posteriorly, finding strong nodes by computing the overall semantic connectivity of each nodes to the rest nodes in a certain context and choosing the top ranked nodes. An alternative strategy can be done similarly as pruning weak nodes, using a prior probability distribution to guide for pruning (see Section 3.4.2). We leave this part open for future work.

3.5 Experimental Results

In addition to NGD, we also experimented with a PMI-based measure of semantic relatedness measure. The original idea of choosing PMI(Section 3.2) as the semantic relatedness function h is to avoid the unstable M problem caused by the search engine (see Section 2.1.3). As the experimental results show in Table 3.1, the performance is in general worse than the baselines (see Section 2.3.2).

Although the performance on the literal class is much worse than the baseline, B_2, the performance on the non-literal class is relatively high. It does not make sense to compare the non-literal performance with B_2, which is very conservative at literal prediction. It identifies literal usage by checking literal repetition of MWE

	N_{output}	L_{output}	$Prec.$	$Rec.$	$F_{\beta=1}$	$Acc.$
N_{label}	2655	447	0.80	0.86	0.83	0.72
L_{label}	645	217	0.33	0.25	0.28	
$Percent.$	0.83	0.17				

Table 3.1: CGA PMI output: output non-literal by the classifier (N_{output}), labeled as literal by annotator (L_{label}), the percentage of total instances output by classifier as literal or non-literal (Percent.)

component words in the certain context, taking the default as predicting non-literal if no repetition is found. The good performance on the non-literal case can be attributed to the imbalanced data, in which non-literal examples are as three times as the literal ones. For the same reason, it is not comparable of the non-literal performance with baseline B_1.

One explanation for the good performance of the *PMI CGA* on the non literal use case might be that the connectivity of the cohesion graph has a tendency to decrease as the nodes of the graph increase, if it is not the truth that the *PMI CGA* is effective at identifying non-literal class. To test whether this guess is true or not, another experiment was done to test how random deletion of nodes affects the overall connectivity of the graph. For every instance, we delete the same number of nodes as the tokens in the MWE. The result (as shown in Table 3.2) shows that the graph does not have a tendency to decrease the connectivity when random words are removed, thus, we believe that the strong bias of predicting non-literal use is not by coincident. The MWE component words encode special cohesion information that can be made use of in our task.

We conducted further experiments, and found out some problems with the *PMI* relatedness function:

$$PMI(x,y) = \log \frac{P(x,y)}{P(x).P(y)} \tag{3.20}$$

where $P(x)$ is the frequency of the term x, $P(x,y)$ is the joint frequency of term x, y.

If we use the page counts returned by search engine to estimate the probability P

	$Con. \uparrow$	$Con. \downarrow$	$Prec.$	$Rec.$	$F_{\beta=1}$	$Acc.$
N_{label}	1436	1666	0.77	0.46	0.58	0.47
L_{label}	428	434	0.21	0.50	0.30	
$Percent.$	0.47	0.53				

Table 3.2: Random deletion of nodes: connectivity increases after random nodes are deleted ($Con. \uparrow$), labeled as literal by annotator (L_{label}), the percentage of total instances of which the connectivity increase or decrease (Percent.)

in Formula 3.20, then it can be rewritten as:

$$PMI(x, y) = \log \frac{M.N(x, y)}{N(x).N(y)} \tag{3.21}$$

where $N(x)$ is the absolute count of term x, $N(x, y)$ is the absolute count of the co-occurrence of the term x, y.

There are two problems with Formula 3.21:

Suppose that if we encounter a case where the joint observation is 0, then Formula 3.21 is underspecified. One simple smoothing technique can be *Add one*[2] (see Formula 3.22).

$$PMI(x, y) = \log \frac{1.M}{N(x).N(y)} \tag{3.22}$$

Then, the problem is that the PMI value of different term pairs, whose joint count is 0, may vary according to the number of the individual counts of the relative term. This result counters intuition, since we would prefer to assign a same small value to all the term pair whose joint count equals to 0, in other words, we need normalization.

The second problem with the definition of PMI is that we get different self-similarity[3] by different words. Formula 3.21 can be rewritten as 3.23 for the self

[2]Add one to the absolute count of the joint observation.

[3]Self-similarity refers to the similarity between a word and itself.

	N_{output}	L_{output}	$Prec.$	$Rec.$	$F_{\beta=1}$	$Acc.$
N_{label}	2518	584	0.90	0.81	0.86	0.79
L_{label}	268	594	0.50	0.69	0.58	

Table 3.3: CGA NGD output

similarity by inputting the same term.

$$PMI(x,x) = \log \frac{M}{N(x)} \tag{3.23}$$

By this definition, the self-similarity value depends on the count of the word itself, which means that the similarity between *teacher and teacher* would be different from the similarity *student and student* if the web counts of *student* is different from *teacher*. This also counters our intuition.

We replaced the definition of the *semantic relatedness* function h with NGD motivated by the fact that it is normalized (see detail [11]). The experimental results are shown in Table 3.3. The NGD method has a much better performance compared with the PMI in both the literal class and the non-literal class.

Motivated by the fact that cohesion-based approach is sensitive to context, we further did experiment to test our graph method with a smaller context: building the graph only based on the current paragraph (see 3.4 for results). Experimental result shows that smaller context is worse in performance. It might be the case that the cohesion graph gets the optimal performance with a certain range of context, but we did not do further experiment to test how CGA responds to context due to a lack of more context in our experiment data[4].

We did some experiments to test our proposed pruning strategy: discarding the weak chain words. We deleted the top-3 bad connected nodes from the graph before classification (see Table 3.5). Experiment results show that it does not really work. One possible reason is that all the nodes in the context are well connected after

[4]All our data instances contain five paragraphs.

	N_{output}	L_{output}	U_{output}	*Prec.*	*Rec.*	$F_{\beta=1}$	*Acc.*
N_{label}	2234	800	68	0.89	0.72	0.80	0.71
L_{label}	274	577	11	0.42	0.67	0.51	

Table 3.4: CGA_p^* output (the cohesion graph is built with the current paragraph instead of the whole context); U_{output} means unpredictable (some instances are not predictable due to the fact that the MWE is the only semantic bearing element of the paragraph)

	N_{output}	L_{output}	*Prec.*	*Rec.*	$F_{\beta=1}$	*Acc.*
N_{label}	2452	650	0.91	0.79	0.85	0.78
L_{label}	240	622	0.49	0.72	0.58	

Table 3.5: Performance of pruning (after the top-3 poorly connected nodes are removed)

the filtering step (see Section 3.4.1), all nodes with close *NGD scores*, which means that even the top-3 bad connected nodes contribute the semantic connectivity of the discourse. Experiment that tried to prune the well-connected nodes got similar result, with a slight drop on performance. This might be a further proof of the evenly distribution of the relatedness score between different nodes in the cohesion graph. We can further study this problem in the future by running some distributional test of the semantic relatedness score of the nodes in the cohesion graph. That may give us more inside view of how to prune the graph.

Table 3.6 shows the comparison of the chain-based approach and cohesion graph-based approach with the baselines. $B2$ is the literal repetition predication based on nouns. LC is the lexical chain-based approach optimized on literal F-score, while LC^* is optimized on accuracy. LC_o is the performance of using the whole date set as the development set (use the the best combination of *relatedness threshold* and *classification threshold*). LC_b the best performance of our data split strategies[5], while LC_w is the worst performance.

The performance of the graph-based approach is as good as the best set of the

[5]Data split deals with how to split data into *development set* and *evaluation set*.

supervised lexical chain classifier.

The graph-based approach is still comparable to the performance of the chain classifier, even when the latter uses the best parameter setting trained on the whole data set.

We think the main reason is that the chaining algorithm is very sensitive to different chain forming strategies (use a greedy search or take a candidate chain pool to keep all the possible chains in order to give rooms to backward context), word-chain relatedness strategy (the new word should be related to *one word* or *all words* in the chain), semantic relatedness threshold (how to get a specific number of the semantic relatedness threshold for a new word to be placed into an existing chain), conflict resolution (how to solve conflict when there are multiple choices for new word to be placed into). Our proposed chaining algorithm is only one different solution out of many alternatives. It is possible that other chaining approaches work better for this specific task.

Furthermore, the classifier also depends on the chain scoring strategy, since it also plays role in the supervised parameter optimization.

All this factors make the lexical chain-based approach very sensitive to experiment setting. Our proposed cohesion graph-based approach avoided the multiple parameters co-effect problem of the chain approach, while maintaining a high performance. It further proves that the cohesion-based approach works with literal or non-literal identification of MWEs.

Method	LPrec.	LRec.	$LF_{\beta=1}$	*Acc.*
$B1$	–	–	–	0.78
$B2$	0.69	0.40	0.50	0.83
$B2^*_p$	0.77	0.19	0.30	0.81
LC_b	0.48	0.77	0.59	0.75
LC_w	0.24	1.00	0.38	0.33
LC_o	0.53	0.69	0.61	0.81
LC^*_o	0.67	0.50	0.57	0.84
CGA^*_p	0.42	0.67	0.51	0.71
CGA	0.5	0.69	0.58	0.79

Table 3.6: Comparison of different approaches: literal repetition on the whole context (B2), literal repetition on the current paragraph ($B2^*_p$), best parameter setting of the chain approach (LC_b), worst parameter setting of the chain approach (LC_w), upper bound performance of the chain approach optimized on literal F-Score (LC_o), upper bound performance of the chain approach optimized on accuracy (LC^*_o), cohesion graph built on the current paragraph (CGA^*_p), cohesion graph built on the whole context (CGA)

Chapter 4

Conclusion and Future Work

4.1 Conclusion

In this thesis, we described a novel approach for token-based idiom classification. Our approach is based on the observation that literally used expressions typically exhibit strong cohesive ties with the surrounding discourse, while idiomatic expressions do not. Idiomatic use of MWEs can be detected by the absence of such ties.

We propose two methods that exploit this behavior. The first method creates lexical chains for the input text and determines the extent to which a potentially idiomatic expression participates in these chains.

The chain-based classifier gained a good performance in our experiments, but it turned out to be sensitive to parameter and data settings. Different *chaining strategy, relatedness threshold, classification threshold* influence the performance a lot. While it is hard to guarantee that our specific chaining algorithm can actually tune the parameter to a global optimal solution, the relatively small experiment data set aggravates the difficulties of setting the parameters even more, as the proportion of development set and evaluation set has an effect on the parameter optimization.

In order to avoid these problems, we proposed a cohesion graph based strategy which compare how the expression component words contribute the overall semantic connectivity of the cohesion graph that is built from the surrounding context. It keeps the idea of making use of lexical cohesion, but avoids the problem of the experimental setup problems of the chaining algorithm. Furthermore, it developed our cohesion-based approach into a fully unsupervised algorithm while maintaining the high performance provided by the supervised chaining approach at the same time.

In conclusion, the main contributions of this work are:

- Implemented the *NGD* semantic relatedness model, tested the stability of search engine, compared the performance of different search engines by detailed technical data. We showed the effectiveness of using distributional approach for modeling semantic relatedness with detailed experiment data. Furthermore, we pointed out that the web-based distributional approach beats the fixed corpus approach, by showing that *NGD* gains better performance than *dependency vector*.

- Proposed a novel cohesion-based approach to identify idiomatic use of expressions. Implement the lexical chain-based approach, and found out the parameter and data sensitivity problems in our application task.

- Proposed an unsupervised graph-based method that aims to make use of lexical cohesion on one hand, while avoids various problems of the chaining algorithm on the other hand. Experimental results show that the graph-based approach keeps a comparable performance as the supervised chaining approach while being unsupervised.

4.2 Future Work

While the cohesion-graph method computes how the idiomatic component words participate in the overall semantic connectivity of the whole context, the chaining approach is more focused on some local cohesion (test how idiomatic component words participate in chains). One interesting point of future work would be to test if local cohesion is better than global cohesion by doing more comparison on the chain-approach and the graph-approach.

The upper bound of our chaining algorithm described in Chapter 2 is based on the specific chaining strategy. We evaluate the chain based on the evaluation on the subsequent task: identifying literal or non-literal use of MWEs. In principle, it is still possible that the performance of our chain-classifier is influenced not only by how lexical cohesion contribute to the idiom identification, but by the quality of the chain itself. We did not make a large scale evaluation on the quality of the chains due to the lack of a gold standard data of lexical chains. But the evaluation based on a subsequent task which skips the evaluation of the chain itself looks still somewhat problematic. We want to do further study on the evaluation of the chains, and we also want to try different chaining algorithms to test whether there are alternatives to improve the local cohesion based approach.

For the graph part, we think it might be very interesting to do more tests on the pruning (see Section 3.4.3). We have suggested that strong cohesion words may lead to a bias on the graph-based classifier, but our experimental result did not find this bias. We think the main reason is that the semantic relatedness scores of the nodes the our experiment data is evenly distributed. We want to do further study to test the statistical distribution of the semantic relatedness of all the edges in the cohesion graph, and then prune the graph based on the statistical distribution properties of the data.

In the future work, we can also explore whether better performance can be achieved by employing a bootstrapping approach, in which we used the examples on which the unsupervised classifier is most confident (i.e., those with the largest difference in connectivity in either direction) as input for a second stage supervised classifier. Whether such a strategy is successful depends to some extent on how well the classifier's confidence correlates with the actual label (i.e., on how well the classifier separates literal and non-literal cases when ranking them). Furthermore, we can also introduce more complex confidence scoring algorithm so that it contains more informative information of the cohesion graph instead of only using connectivity gain.

Appendix A

Lexical Chain Samples

****

⟨idiom=break the ice⟩ ⟨file=afe199410⟩ ⟨label=n⟩

Holst, who had twice served as defense minister, took over the Norwegian foreign ministry in 1993 when his predecessor, Thorvald Stoltenberg, left to work as the UN mediator in the Yugoslav war.

His secret talks with representatives from Israel and the Palestine Liberation Organization started almost immediately. Though he did not initiate the Middle East dialogue, it was his work in pressing ahead that allowed the breakthrough.

With his wife Marianne Heiberg, he used the family setting of his Oslo home to help *break the ice* in the early contacts between the Israelis and the PLO.

"Conviviality was at least as important as secrecy in this peace process," he said afterwards.

A tireless traveler, he made numerous trips late last year to the Middle East and other world capitals to further the peace process. But the hectic pace proved too much and Holst died in his sleep in January at age 56 after suffering two strokes – only four months after the signing of the historic accord.

****

Lexical Chains

holst Thorvald Stoltenberg Heiberg Holst

twice secret almost immediately though ahead late proved

defense foreign representatives

minister ministry peace peace suffering

norwegian Oslo

1993

predecessor

mediator initiate

yugoslav Israel

Palestine Liberation Israelis Plo

middle middle

dialogue accord

breakthrough

wife afterwards

Marianne

setting

* ice *

Conviviality

secrecy

tireless

traveler trips

numerous further

capitals

hectic

pace

sleep strokes

signing

historic

****

⟨idiom=break the ice*rlangle* ⟨file=afe199412⟩ ⟨label=n⟩

Syria said Wednesday that peace with Israel could only be reached "on the ground" and not just on paper, after US Secretary of State Warren Christopher took his peace shuttle to Israel.

Christopher tried to "*break the ice* and narrow the gap" between Syria and Israel during "intensive" talks on Monday with President Hafez al-Assad, the English-language Syria Times reported.

"His job, however, is not easy considering the most recently declared Israeli positions," the official daily said.

Quoting recent remarks by Assad it said "peace must be just and comprehensive on the ground. In other words, peace canot be made on paper. It must be realised on the ground to ensure its durability and its effectiveness."
****

Lexical Chains

Syria peace Israel peace Israel Syria Israel Syria Israeli peace peace

Wednesday Monday

reached ground ground ground

secretary declared remarks

Warren Christopher Christopher

shuttle

* ice *

narrow

gap effectiveness

intensive canot

Hafez Al-assad assad

english-language

recently

positions ensure

comprehensive

realised

durability

****

⟨idiom=break the ice⟩ ⟨file=apw199808⟩ ⟨label=l⟩

Continents, said Hoffman, probably were in a dry, cold soak.

"Once the seas froze over, there was no more evaporation," he said. "There was no more snow or rain." Glaciers on land disappeared. Continents became like dry and lifeless rocks in frozen seas.

The ancient ice ages ended when carbon dioxide, belched from volcanoes, became concentrated enough in the atmosphere about 350 times the present concentration to create a super greenhouse effect. The carbon dioxide trapped enough solar heat to melt the frozen oceans and to *break the ice* age.

The Earth went through this cycle repeatedly as the continents drifted apart, Hoffman said. But such severe ice ages are unlikely to happen again for two reasons: the sun is about 7 percent hotter, and higher life forms continuously cycle carbon back into the atmosphere, maintaining a gas blanket that warms the planet.

"Our friends the worms and snails keep this kind of ice event from happening now," Hoffman said. "They scavenge the organic matter on the sea floor and recycle it. There was no way to have this high a rate of organic burial once higher animals evolved."

****

Lexcial Chains

continents continents ancient continents apart

Hoffman Hoffman Hoffman

probably cold enough enough

dry soak evaporation dry melt

seas seas oceans sea

froze snow frozen ice frozen * ice * ice ice

rain heat severe warms

glaciers dioxide volcanoes greenhouse dioxide

disappeared concentrated concentration unlikely evolved

lifeless belched

carbon carbon solar carbon organic recycle organic

atmosphere atmosphere

350

present effect higher maintaining matter higher

trapped

earth planet

cycle cycle

repeatedly continuously

drifted

percent

hotter

gas

blanket

worms snails

scavenge

floor

burial

animals

Appendix B

Cohesion Graph Samples

B.1 Experimental Results

⟨record idiom=”[2:0:9_biting_VVG , 2:0:10_off_RP , 2:0:11_more_DAR , 2:0:12_than_CSN , 2:0:13_you_PPY , 2:0:14_can_VM , 2:0:15_chew_VV0]” id=”46” label=”n” baseline=”n” output=” n” combine=”n” ⟩
⟨graph1 vex=”50” edge=”1225” con=”0.7533231871419971” /⟩
⟨graph2 vex=”48” edge=”1128” con=”0.7503445073592749” /⟩
⟨/record⟩
⟨record idiom=”[2:0:2_bit_VVD , 2:0:3_off_RP , 2:0:4_more_DAR , 2:0:5_than_CSN , 2:0:6_they_PPHS2 , 2:0:7_could_VM , 2:0:8_chew_VV0]” id=”47” label=”n” baseline=”n” output=”n” c ombine=”n” ⟩
⟨graph1 vex=”32” edge=”496” con=”0.8662333498944009” /⟩
⟨graph2 vex=”30” edge=”435” con=”0.86265609908176” /⟩
⟨/record⟩
⟨record idiom=”[2:0:2_bit_VVD , 2:0:3_off_RP , 2:0:4_more_DAR , 2:0:5_than_CSN , 2:0:6_they_PPHS2 , 2:0:7_could_VM , 2:0:8_chew_VV0]” id=”48” label=”n” baseline=”n” output=”n” c ombine=”n” ⟩
⟨graph1 vex=”65” edge=”2080” con=”0.8246161194671754” /⟩
⟨graph2 vex=”63” edge=”1953” con=”0.8225377551398092” /⟩
⟨/record⟩

⟨record idiom="[2:0:2_bit_VVD , 2:0:3_off_RP , 2:0:4_more_DAR , 2:0:5_than_CSN , 2:0:6_they_PPHS2 , 2:0:7_could_VM , 2:0:8_chew_VV0]" id="49" label="n" baseline="n" output="n" c ombine="n" ⟩
⟨graph1 vex="65" edge="2080" con="0.8244317974444197" /⟩
⟨graph2 vex="63" edge="1953" con="0.8222542813812084" /⟩
⟨/record⟩
⟨record idiom="[2:1:8_biting_VVG , 2:1:9_off_RP , 2:1:10_more_DAR , 2:1:11_than_CSN , 2:1:12_they_PPHS2 , 2:1:13_can_VM , 2:1:14_chew_VV0]" id="50" label="l" baseline="n" output ="n" combine="n" ⟩
⟨graph1 vex="84" edge="3486" con="0.7459496850391863" /⟩
⟨graph2 vex="82" edge="3321" con="0.7450319533485049" /⟩
⟨/record⟩
⟨record idiom="[2:2:36_biting_VVG , 2:2:37_off_RP , 2:2:38_more_DAR , 2:2:39_than_CSN , 2:2:40_you_PPY , 2:2:41_can_VM , 2:2:42_chew_VV0]" id="51" label="n" baseline="n" output= "l" combine="l" ⟩
⟨graph1 vex="78" edge="3003" con="0.7737341929972239" /⟩
⟨graph2 vex="76" edge="2850" con="0.7744978322303094" /⟩
⟨/record⟩
⟨record idiom="[2:0:16_bit_VVD , 2:0:17_off_RP , 2:0:18_more_DAR , 2:0:19_than_CSN , 2:0:20_he_PPHS1 , 2:0:21_could_VM , 2:0:22_chew_VV0]" id="52" label="n" baseline="n" output= "n" combine="n" ⟩
⟨graph1 vex="78" edge="3003" con="0.7586503709415167" /⟩
⟨graph2 vex="76" edge="2850" con="0.7561728091326653" /⟩
⟨/record⟩

B.2 Graph Connectivity

⟨record idiom="[2:0:33_backed_VVN , 2:0:34_the_AT , 2:0:35_wrong_JJ , 2:0:36_horse_NN1]" id="2" label="n" output="l" ⟩

```
⟨graph1 vex=”43” edge=”903” con=”0.8211514227989734” /⟩
⟨graph2 vex=”41” edge=”820” con=”0.8228523849909961” /⟩
⟨keyedge⟩
(2:0:35_wrong_JJ ,0:0:29_drugs_NN2 ,0.6460493499788731)
(2:0:36_horse_NN1 ,2:0:6_esteemed_VVN ,0.7861418321109883)
(2:0:35_wrong_JJ ,0:0:5_systematic_JJ ,0.7486807368628665)
(2:0:35_wrong_JJ ,1:0:4_Mustapha_NP1 ,0.9096717663572721)
(2:0:36_horse_NN1 ,1:0:2_federation_NN1 ,0.9330524282320959)
(2:0:36_horse_NN1 ,0:0:9_Yoshio_NP1 ,1.0013374493101597)
(2:0:35_wrong_JJ ,0:0:7_programme_NN1 ,0.9626915893949558)
(2:0:36_horse_NN1 ,0:0:16_council_NNJ1 ,0.8343561996288718)
(2:0:36_horse_NN1 ,0:0:33_coaches_NN2 ,0.6310848368316496)
(2:0:35_wrong_JJ ,1:0:6_claims_VVZ ,0.5635336094746174)
(2:0:36_horse_NN1 ,2:0:7_british_JJ ,0.7474692379327043)
(2:0:36_horse_NN1 ,2:0:8_coach_NN1 ,0.6310848368316496)
(2:0:36_horse_NN1 ,0:0:3_denied_VVN ,0.7259803655625995)
(2:0:36_horse_NN1 ,0:0:6_drug_NN1 ,0.8014239376204161)
(2:0:35_wrong_JJ ,2:0:29_1990_MC ,0.9575873415424562)
(2:0:35_wrong_JJ ,0:0:3_denied_VVN ,0.5650184662600951)
(2:0:36_horse_NN1 ,0:0:29_drugs_NN2 ,0.8014239376204161)
(2:0:35_wrong_JJ ,2:0:7_british_JJ ,0.7240540403798668)
(2:0:35_wrong_JJ ,0:0:6_drug_NN1 ,0.6460493499788731)
(2:0:36_horse_NN1 ,1:0:6_claims_VVZ ,0.7556251333493157)
(2:0:36_horse_NN1 ,1:0:8_integrity_NN1 ,0.827016524514288)
(2:0:35_wrong_JJ ,0:0:28_high-tech_NN1 ,0.8575828234559612)
(2:0:36_horse_NN1 ,1:0:3_chief_NN1 ,0.7330445361266866)
(2:0:36_horse_NN1 ,0:0:31_swimmers_NN2 ,0.775222688065663)
(2:0:35_wrong_JJ ,0:0:36_formidable_JJ ,0.7542835304326948)
(2:0:36_horse_NN1 ,0:0:21_commission_NN1 ,0.8682415987331692)
(2:0:35_wrong_JJ ,1:0:1_swimming_NN1 ,0.842994052586938)
```

```
(2:0:36_horse_NN1 ,0:0:28_high-tech_NN1 ,0.9376652088464912)
(2:0:36_horse_NN1 ,2:0:27_Beijing_NP1 ,0.8558015693136459)
(2:0:36_horse_NN1 ,2:0:18_german_JJ ,0.8176188972047927)
(2:0:36_horse_NN1 ,2:0:2_Dave_NP1 ,0.7434248123279089)
(2:0:35_wrong_JJ ,2:0:8_coach_NN1 ,0.7979767426715284)
(2:0:35_wrong_JJ ,1:0:17_scandal_NN1 ,0.6504160552020153)
(2:0:35_wrong_JJ ,2:0:6_esteemed_VVN ,0.6900619832484323)
(2:0:35_wrong_JJ ,0:0:33_coaches_NN2 ,0.7979767426715284)
(2:0:35_wrong_JJ ,0:0:21_commission_NN1 ,0.7671706593201756)
(2:0:36_horse_NN1 ,0:0:10_kuroda_NN1 ,0.8300973735843992)
(2:0:35_wrong_JJ ,1:0:2_federation_NN1 ,0.9163533131773389)
(2:0:35_wrong_JJ ,2:0:18_german_JJ ,0.7487403025280962)
(2:0:35_wrong_JJ ,0:0:20_medical_JJ ,0.8095767233879482)
(2:0:36_horse_NN1 ,2:0:24_asian_JJ ,0.8235640573424865)
(2:0:35_wrong_JJ ,1:0:3_chief_NN1 ,0.6621172818200607)
(2:0:36_horse_NN1 ,0:0:5_systematic_JJ ,0.9063062261671875)
(2:0:35_wrong_JJ ,2:0:11_accused_VVN ,0.6139163597449714)
(2:0:36_horse_NN1 ,0:0:15_olympic_JJ ,0.5765473322859024)
(2:0:36_horse_NN1 ,2:0:29_1990_MC ,0.9865726318330111)
(2:0:35_wrong_JJ ,0:0:16_council_NNJ1 ,0.7525028356874036)
(2:0:36_horse_NN1 ,0:0:1_chinese_NN2 ,0.8709241206206583)
(2:0:36_horse_NN1 ,0:0:36_formidable_JJ ,0.7936164702002954)
(2:0:35_wrong_JJ ,1:0:12_championships_NN2 ,0.8558910419949107)
(2:0:35_wrong_JJ ,2:0:3_Haller_NP1 ,1.0041378124275326)
(2:0:35_wrong_JJ ,0:0:9_Yoshio_NP1 ,0.9865929606544165)
(2:0:36_horse_NN1 ,1:0:1_swimming_NN1 ,0.6470168669670217)
(2:0:44_racism_NN1 ,2:0:35_wrong_JJ ,0.6442805621492382)
(2:0:35_wrong_JJ ,1:0:8_integrity_NN1 ,0.6868268935528039)
(2:0:36_horse_NN1 ,1:0:17_scandal_NN1 ,0.7575035520380472)
(2:0:35_wrong_JJ ,0:0:10_kuroda_NN1 ,0.952147192172133)
```

```
(2:0:36_horse_NN1 ,1:0:4_Mustapha_NP1 ,0.9490243737347517)
(2:0:35_wrong_JJ ,0:0:15_olympic_JJ ,0.8452277293911455)
(2:0:38_accusing_VVG ,2:0:36_horse_NN1 ,0.7552189919553047)
(2:0:36_horse_NN1 ,0:0:20_medical_JJ ,0.8905762001072248)
(2:0:36_horse_NN1 ,0:0:7_programme_NN1 ,1.0326233138955554)
(2:0:42_jealousy_NN1 ,2:0:36_horse_NN1 ,0.7540394838999024)
(2:0:44_racism_NN1 ,2:0:36_horse_NN1 ,0.787391600134764)
(2:0:36_horse_NN1 ,1:0:5_Larfaoui_NP1 ,1.033924559070483)
(2:0:36_horse_NN1 ,2:0:31_Larfaoui_NP1 ,1.033924559070483)
(2:0:36_horse_NN1 ,2:0:3_Haller_NP1 ,0.9836213163576962)
(2:0:36_horse_NN1 ,1:0:14_untouched_JJ ,0.7741433090397059)
(2:0:35_wrong_JJ ,2:0:27_Beijing_NP1 ,0.8884541218270144)
(2:0:35_wrong_JJ ,1:0:14_untouched_JJ ,0.7413262790455174)
(2:0:36_horse_NN1 ,1:0:12_championships_NN2 ,0.6133535865094725)
(2:0:36_horse_NN1 ,0:0:18_Asia_NP1 ,0.9281944231142875)
(2:0:35_wrong_JJ ,0:0:31_swimmers_NN2 ,0.8347295197996393)
(2:0:35_wrong_JJ ,0:0:18_Asia_NP1 ,0.853996523294991)
(2:0:35_wrong_JJ ,1:0:5_Larfaoui_NP1 ,1.0358697218179613)
(2:0:35_wrong_JJ ,2:0:2_Dave_NP1 ,0.691057977358222)
(2:0:35_wrong_JJ ,2:0:31_Larfaoui_NP1 ,1.0358697218179613)
(2:0:36_horse_NN1 ,2:0:35_wrong_JJ ,0.7162719752196649)
(2:0:35_wrong_JJ ,0:0:1_chinese_NN2 ,0.8088752385784955)
(2:0:35_wrong_JJ ,2:0:24_asian_JJ ,0.8095110312320719)
(2:0:42_jealousy_NN1 ,2:0:35_wrong_JJ ,0.6757486528108313)
(2:0:38_accusing_VVG ,2:0:35_wrong_JJ ,0.6130267330581117)
(2:0:36_horse_NN1 ,2:0:11_accused_VVN ,0.760731372396069)
⟨/keyedge⟩
⟨/record⟩
```

Bibliography

[1] Timothy Baldwin, Emily M. Bender, Dan Flickinger, Ara Kim, and Stephan Oepen. Road-testing the english resource grammar over the british national corpus. In *In Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC 2004)*, pages 2047–2050, 2004.

[2] Timothy Baldwin and Aline Villavicencio. Extracting the unextractable: A case study on verb-particles. In *Proceedings of the Sixth Conference on Computational Natural Language Learning (CoNLL 2002)*, pages 98–104, Taipei, Taiwan, 2002.

[3] Colin Bannard, Buccleuch Place, Timothy Baldwin, and Alex Lascarides. A statistical approach to the semantics of verb-particles, May 26 2003.

[4] R. Barzilay and M. Elhadad. Using lexical chains for summarization. In *ACL/EACL-97 summarization workshop*, pages 10–18, Madrid, Spain, 1997.

[5] Julia Birke and Anoop Sarkar. A clustering approach for nearly unsupervised recognition of nonliteral language. In *EACL-2006*, Trento, Italy, 2006.

[6] Francis Bond and Satoshi Shirai. Practical and efficient organization of a large valency dictionary. In *In Workshop on Multilingual Information Processing Natural Language Processing Pacific Rim Symposium 97: NLPRS-97*, 1997.

[7] Ted Briscoe, John Carroll, and Rebecca Watson. The second release of the RASP system. In *ACL*. The Association for Computer Linguistics, 2006.

[8] Meru Brunn, Yllias Chali, and Christopher J. Pinchak. Text summarization using lexical chains, September 08 2001.

[9] Yllias Chali and Shafiq R. Joty. Uofl: Word sense disambiguation using lexical cohesion. In *Proceedings of the 4th International Workshop on Semantic Evaluations (SemEval-2007)*, pages 476–479, Prague, 2007.

[10] W. Chapman. *Roget's International Thesaurus.* Collins, Publ., London, 1988.

[11] Cilibrasi and Viltanyi. The google similarity distance. *IEEETKDE: IEEE Transactions on Knowledge and Data Engineering*, 19, 2007.

[12] William Doran, Nicola Stokes, Joe Carthy, and John Dunnion. Comparing lexical chain-based summarisation approaches using an extrinsic evaluation, March 09 2004.

[13] Stefan Evert and Brigitte Krenn. Methods for the qualitative evaluation of lexical association measures. In *ACL*, pages 188–195, 2001.

[14] Afsaneh Fazly and Paul Cook. Unsupervised type and token identification of idiomatic expressions. *Computational Linguistics (to appear).*

[15] Laurie Gerber and Jin Yang. Systran mt dictionary development. In *In Proceedings of the Fifth Machine Translation Summit (MT Summit V)*, San Diego, USA, 1997.

[16] Graeme Hirst and David St-Onge. Lexical chains as representations of context for detection and correction of malapropisms. In Christaine Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 305–332. The MIT Press, Cambridge, Massachusetts, 1998.

[17] Ray Jackendoff, Samuel Jay Keyser, and Suzanne Stevenson. Review of: The architecture of the language faculty , ray jackendoff (brandeis university) cambridge, MA: The MIT press (linguistic inquiry monographs, edited by samuel jay keyser, volume 28), 1997, xvi+262 pp; paperbound, ISBN 0-262-60025-0, June 11 1997.

[18] Mario Jarmasz and Stan Szpakowicz. Not as easy as it seems: Automating the construction of lexical chains using roget's thesaurus. In Yang Xiang and Brahim Chaib-draa, editors, *Canadian Conference on AI*, volume 2671 of *Lecture Notes in Computer Science*, pages 544–549. Springer, 2003.

[19] Graham Katz and Eugenie Giesbrecht. Atomatic identification of non-compositional multi-word expressions using latent semantic analysis. In *Proceedings of the ACL-06 Workshop on Multiword Expressions: Identifying and Exploiting Underlying Properties*, Sydney, Australia, 2006.

[20] M. Kendall. *A New Measure of Rank Correlation*, volume 30, pages 81–89. Biometrika, 1938.

[21] Mirella Lapata and Frank Keller. Web-based models for natural language processing. *TSLP*, 2(1):1–31, 2005.

[22] David D. Lewis and W. Bruce Croft. Term clustering of syntactic phrases. In *Proceedings of SIGIR-90, 13th ACM International Conference on Research and Development in Information Retrieval*, pages 385–404, Bruxelles, BE, 1990.

[23] Dekang Lin. Using collocation statistics in information extraction. In *In Proceedings of the Seventh Message Understanding Conference (MUC-7)*, 1998.

[24] Dekang Lin. Automatic identification of non-compositional phrases. In *ACL*, 1999.

[25] Rila Mandala, Takenobu Tokunaga, and Hozumi Tanaka. Query expansion using heterogeneous thesauri. *Inf. Process. Manage*, 36(3):361–378, 2000.

[26] J. Morris and G. Hirst. Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics*, 17(1):21–43, 1991.

[27] Jerome L. Myers and Arnold D. Well. *Research Design and Statistical Analysis.* Lawrence Erlbaum, second edition, 2003.

[28] Manabu Okumura and Takeo Honda. Word sense disambiguation and text segmentation based on lexical cohesion. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 2, pages 755–761. COLING, 1994.

[29] S. Pado and M. Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2), 2007.

[30] Sebastian Pado and Mirella Lapata. Constructing semantic space models from parsed corpora. In *Proceedings of ACL-03*, Sapporo, Japan, 2003.

[31] Piao, Scott S.L., Paul Rayson, Dawn Archer, Andrew Wilson, and Tony Mcenery. Extracting multiword word expressions with a semantic tagger. In *In Proc. of the ACL-2003 Workshop on Multiword Expression: Analysis, Acquisition and Treatment*, pages 49–56, Sapporo, japan, 2003.

[32] Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. Multiword expressions: A pain in the neck for NLP. *Lecture Notes in Computer Science*, 2276:1–29, 2001.

[33] G. Salton, C Buckley, and A. Smith. On the applicability of syntactic methodologies in automatic text retrieval. *Information Processing and Management*, 26(1), 1990.

[34] H. Gregory Silber and Kathleen F. McCoy. Efficient text summarization using lexical chains. In *Intelligent User Interfaces*, pages 252–255, 2000.

[35] H. Gregory Silber and Kathleen F. McCoy. Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics*, 28(4):487–496, 2002.

[36] Frank Smadja. Retrieving collocations from text: Xtract. *Computational Linguistics*, 33:143–77, 1993.

[37] David St-onge. Detecting and correcting malapropisms with lexical chains. Technical report, March 31 1995.

[38] Elke Teich and Peter Fankhauser. Wordnet for lexical cohesion analysis. In *Proceedings of the Second Global Wordnet Conference*, pages 326–331, Brno, Czech Republic, 2004.

[39] Randee I. Tengi. Design and implemetation of the WordNet database and searching software. In Christaine Fellbaum, editor, *WordNet: An Electronic Lexical Database*, pages 105–127. The MIT Press, Cambridge, Massachusetts, 1998.

[40] Christoph Tillmann and Hermann Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133, 2003.

[41] Nina Wacholder and Peng Song. Toward a task-based gold standard for evaluation of NP chunks and technical terms. In *HLT-NAACL*, 2003.

[42] Xiaojin Zhu and Ronald Rosenfeld. Improving trigram language modeling with the world wide web.