NAIST-IS-DD0761015

Doctoral Dissertation

Graph-Theoretic Approaches to Minimally-Supervised Natural Language Learning

Mamoru Komachi

March 17, 2010

Department of Information Processing
Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Mamoru Komachi

Thesis Committee:

Professor Yuji Matsumoto	(Supervisor)
Professor Hiroyuki Seki	(Co-supervisor)
Associate Professor Kentaro Inui	(Co-supervisor)
Assistant Professor Masashi Shimbo	(Co-supervisor)
Assistant Professor Patrick Pantel	(University of Southern California/ISI)

Graph-Theoretic Approaches to Minimally-Supervised Natural Language Learning*

Mamoru Komachi

Abstract

Bootstrapping is a minimally supervised machine learning algorithm used in natural language processing (NLP) to reduce the cost of human annotation. It starts from a small set of seed instances (e.g., (cat, animal) for learning is-a relation) to extract context patterns (e.g., "X such as Y") from a corpus. The extracted patterns are used to extract other target instances which co-occur with the patterns, and the extracted instances are then used for inducing other context patterns. By applying these steps iteratively, one can easily multiply the number of seed instances with minimal human annotation cost. The idea of bootstrapping has been adopted to many NLP tasks such as relation extraction and named entity recognition. However, bootstrapping has a tendency, called *semantic drift*, to select instances unrelated to the seed instances as the iteration proceeds.

The main contribution of this thesis is to demonstrate the semantic drift of bootstrapping has the same root as the topic drift of Kleinberg's HITS, using a simplified graph-based reformulation of bootstrapping. We confirm that two graph-based algorithms, the von Neumann kernels and the regularized Laplacian, can reduce semantic drift in various natural language processing tasks. Proposed algorithms achieve superior performance to *Espresso*, even though the proposed algorithms have less parameters than Espresso and are easy to calibrate.

In this thesis, we first overview bootstrapping algorithms including state-of-the-art bootstrapping algorithm called Espresso with experimental results. We show that Espresso outperforms previous bootstrapping methods, and apply three modifications to the Espresso algorithm to induce a new Espresso-style algorithm called Tchai. Then we present a graph-based analysis of

*Doctoral Dissertation, Department of Information Processing, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DD0761015, March 17, 2010.

Espresso-style bootstrapping algorithms to show the parallel between topic drift and semantic drift, and propose two graph-based algorithms. Finally, we apply the proposed algorithms to three NLP tasks: word sense disambiguation, bilingual dictionary construction and semantic category acquisition. Experimental results show that the regularized Laplacian, one of the proposed algorithms, is comparable to Espresso, yet the former is easy to calibrate and scalable to large scale data.

Keywords:

natural language processing, bootstrapping, link analysis, semi-supervised learning

グラフ理論的観点からの自然言語処理における弱教師あり学習*

小町 守

内容梗概

自然言語処理におけるブートストラッピング手法は，人手による資源作成のコストを減らすための弱教師あり学習の一つである．ブートストラッピング手法は，獲得対象となるクラス，たとえば is-a 関係の少量のインスタンス（例 (cat, animal) ）をシードとして与え，コーパスからインスタンスと共起するパターン（例 “X such as Y”）を抽出する．抽出された共起パターンは，そのパターンと共起する別のインスタンスを獲得するために用いられる．このようなステップを反復的に適用することで，人手による介入を最小限にシードインスタンスの数を簡単に増やすことができる．ブートストラッピング手法の考え方は関係抽出や固有表現認識などさまざまな自然言語処理のタスクに応用されている．しかしながら，ブートストラッピング手法には，反復が進むにつれシードインスタンスとは無関係なインスタンスを抽出してしまう，意味ドリフトと呼ばれる傾向がある．

本研究の主な貢献は，Espresso 型のブートストラッピング手法における意味ドリフト現象が，Kleinberg の HITS と同じ原因によって引き起こされているのだという事実を，グラフ理論を用いて定式化し，単純化されたブートストラッピング手法を用いて解析的に示したことである．また，グラフ理論に基づく 2 つのアルゴリズム（von Neumann カーネルと正則化ラプラシアン）が意味ドリフトの影響を軽減することを，いくつかの自然言語処理タスクによって検証した．提案手法は Espresso より高い精度を達成し，さらに，先行研究より設定するべきパラメータ数が少ないため，調整が容易である．

本論文ではまずブートストラッピング手法について概観する．Espresso という最新のブートストラッピング手法について詳述し，実験的な性能を評価する．Espresso は他のブートストラッピング手法より高い性能であることを示し，Espresso に対して 3 つの修正を行なうことでさらに性能が高い Tchai という Espresso 型の新しいアルゴリズムを導く．そして Espresso 型のブートストラッピングアルゴリズムに対してグラフ理論に基づく分析を行ない，トピックドリフ

*奈良先端科学技術大学院大学 情報科学研究科 情報処理学専攻 博士論文，NAIST-IS-DD0761015, 2010 年 3 月 17 日.

トと意味ドリフトの関係について示し，2つのグラフに基づく手法を提案する．最後に提案手法を3つの自然言語処理タスクに適用した結果について報告する：語義曖昧性解消，対訳辞書構築，そして意味カテゴリ学習である．実験結果によって，提案手法の一つである正則化ラプラシアンは Espresso と同等の性能を持ち，調整が簡単で，大規模データにもスケーラブルであることを示す．

キーワード

自然言語処理，ブートストラッピング，リンク解析，半教師あり学習

Acknowledgements

I am especially fortunate to have been co-supervised by Dr. Patrick Pantel (Yahoo! Labs / University of Southern California). I devoted my dissertation to giving theoretical and empirical analysis of his famous bootstrapping algorithm, called *Espresso*. The work in this thesis was shaped by his great insights. I am very happy to have numerous discussions with him.

(For the rest of the acknowledgement I use Japanese to express my gratitude.)

まず修士から博士まで5年間指導してくださった松本裕治先生に感謝いたします．あちこちインターンシップや共同研究で飛び回っている自分に小言の一つも言わず，自由に研究させてくださり，研究者としての幅を広げることができました．それでも松本先生ほど懐の広い研究者になるにはまだまだかかりそうです．いつの日か松本先生のようになるのが自分の夢です．

次に博士に進学してからずっとお世話になった新保仁さんにお礼申し上げます．毎回論文をチェックしてもらう度に真っ赤になって返ってきて，20往復くらいしてようやく投稿にこぎ着けるまで，長い道のりでしたがとても勉強になりました．「もうこれ以上よくならないだろう」と思うところからさらに真っ赤になるので，研究の奥深さを知りました．今後ともどうぞよろしくお願いします．

また，修士論文を根気強く指導してくださった乾健太郎先生からは，自然言語処理の研究のイロハを教えてくださいました．博士の研究にも毎回本質的なコメントをいただき，世界トップレベルの研究をするにはどう発想するべきか学びました．これからの自然言語処理を盛り立てていきたいと思います．

そして関浩之先生はお忙しい中博士論文の審査を引き受けてくださいました．中間審査で本研究がおもしろいと言ってくださったのが初めて分野外の人から研究をおもしろいと言ってもらった経験であり，それがこれまで研究を続ける原動

力になったと言っても過言ではありません．博士論文も丁寧に見てくださり，どうもありがとうございました．

Microsoft Research の鈴木久美さんには，世界最高峰の研究環境の中で，コンスタントにハイレベルな研究をするためにはどうすればいいのか，教わりました．初めてのアメリカ生活，研究漬けの環境もカルチャーショックでした．検索クエリログという，生のデータを使ってみる，というところから，博士論文につながる研究テーマを見出すことができました．帰国してから初めての論文誌にも投稿し，研究者として生きていけるという自信にもつながりました．研究者の卵として接してくださって，ありがとうございます．

Google の工藤拓さんには，公私ともに計り知れないほどお世話になりました．論文を書くというのはどのようにすればいいのか，初めて書き方を教えてくださったときは,「こうすればいいのか！」と鳥肌が立ちました．まだまだ未熟なので至らないところはあるかと思いますが，今後とも叱咤してくださるとありがたいです．

Yahoo! JAPAN 研究所の颯々野学さんは，六本木の東京ミッドタウンでの 3ヶ月間の共同研究に招いてくださり，大企業の中で研究を仕事にするというのはどういうことかについて，丁寧に教えてくださって，とても勉強になりました．将来どのようなキャリアを辿るか，一生でいちばん考えた時期だと思います．共同研究を手伝ってくれた内海慶さんと牧本慎平くんにも大変感謝しています．どうもお世話になりました．

NTT 研究所の永田昌明さんには，修士のころから数えるとほぼ大学院生活全部に渡ってお世話になりっぱなしで，博士の研究にストーリーを持たせることの重要性，そして研究の作法についてたくさん教えてくださいました．どうもありがとうございました．今後ともどうぞよろしくお願いします．

ひとつの研究テーマにはなりませんでしたが，奈良先端大の支援を受けて大規模なウェブデータを用いた統計的かな漢字変換エンジンの開発に取り組めたのは大きな経験になりました．メンターを引き受けてくださった浅原正幸助教，一緒に議論・開発してくださった徳永拓之さん，森信介さん，アドバイスをくださった高岡一馬さん，中川哲治さん，どうもありがとうございました．そして，Apple Inc. でのインターンシップを通じて，いかに世界最高水準のプロダクトを作っていくか，という姿を教えてくれた木田泰夫さんにも感謝しております．シリコンバレーでの生活は，長い博士の 3 年間における心のオアシスでした．

松本研の卒業生の方々にもお世話になりました．山下達雄さん，高村大也さん，持橋大地さん，乾孝司さん，藤田篤さん，鈴木潤さん，賀沢秀人さん，飯田龍さん，伊藤敬彦さん，小林のぞみさん，平野徹さん，鄭育昌さん，お名前を挙げるときりがないですが，研究者としてどのようにあるべきかアドバイスをいただいたり，励ましていただいたので，ともすれば孤独になりがちな博士の学生生活を乗り切ることができました．どうもありがとうございます．

歴代の研究室秘書の方々，大川基子さんと北川祐子さんには，物品の購入や出張手続きなど煩雑な処理を一手に引き受けてくださって，非常に助かりました．特に北川さんは博士後期課程の 2 年目から，日本学術振興会特別研究員の手続きや研究費の処理でたくさんの仕事をお願いし，本当に頭が上がりません．研究のことに集中できたのも，大川さんと北川さんが事務手続きに留まらず，休みの日の過ごし方や将来のことまで親身になって相談に乗ってくださったおかげだと思います．素敵な方々に囲まれた生活ができてとても幸せでした．

在学生の方々にも大変お世話になっていますが，特に同時に博士を取得する Eric Nichols さん，阿部修也さん，渡邉陽太郎くん，木村学くんに感謝します．ひとりだったら乗り切れなかったと思います．今年みんなで一緒に卒業できてとても嬉しいです．

そして，大学こそ違いますが，岡野原大輔くんと萩原正人さんにも感謝の気持ちを伝えたいです．同世代でこれだけ優れた人たちと一緒に仕事ができるというのは，とても刺激的でした．今後とも「若手」の力を存分に発揮して，分野や大学，そして国の壁を越えて活躍しましょう．

最後になりますが，30 年間に渡る学生生活を支えてくれた両親の寛，久美江，兄弟の順，学，進に感謝します．のんびり学生でいさせてくれたおかげで，こうやって今の自分があるのだと思います．

また，妻の美由紀には遠距離恋愛・遠距離結婚でとてもつらい思いをさせてしまい，申し訳ないです．でも，奈良で一緒に生活した 1 年，人生でいちばん楽しかったです．一生の宝物だと思っています．愛しているよ．またふたりで生駒に来ようね．

Contents

List of Figures

List of Tables

1 Introduction

In recent years machine learning techniques have become widely used in natural language processing (NLP). These techniques offer various ways to exploit large corpora and are known to perform well in many tasks. However, these techniques often require tagged corpora, which are not readily available to many languages. Reducing the cost of human annotation is one of the important factors for building NLP systems. Consequently, extraction of lexical knowledge from a large collection of text data with minimal supervision has been an active area of research recently.

To mitigate the problem of hand-tagging resources, semi(or minimally)-supervised and unsupervised techniques have been actively studied. Automatic extraction of relations by exploiting recurring patterns in text was pioneered by Hearst (1992)[23], who describes a bootstrapping procedure for extracting words in the hyponym (is-a) relation, starting with three manually given lexico-syntactic patterns. Bootstrapping algorithms can easily multiply the number of tagged instances with minimal human annotation cost, by iteratively applying pattern extraction and instance extraction.

This idea of learning with a minimally supervised bootstrapping method was subsequently adopted for many tasks, including word sense disambiguation [61], relation extraction [9, 47, 42] and named entity recognition [14, 20].

However, it is known that bootstrapping often acquires instances not related to seed instances. For example, consider the task of collecting the names of common tourist sites from web corpora. Given words like "Geneva" and "Bali" as seed instances, bootstrapping would eventually learn *generic patterns* such as "pictures" and "photos," which also co-occur with many other unrelated instances. The subsequent iterations would likely acquire frequent

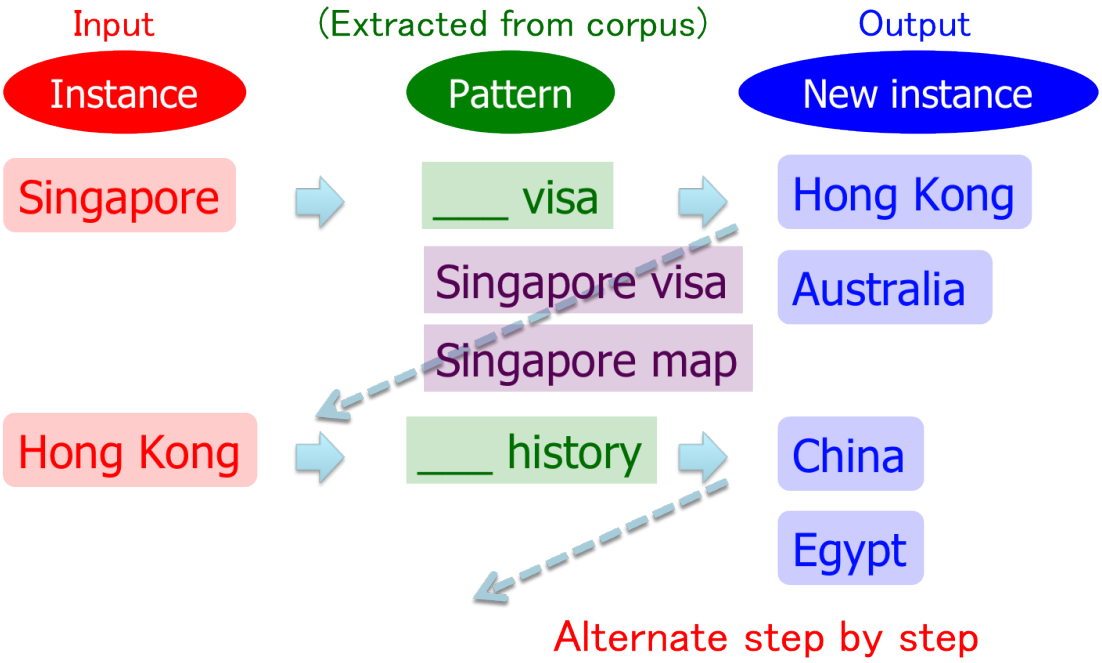

Figure 1.1. Corpus-based extraction of semantic knowledge

words that co-occur with these generic patterns, such as "Britney Spears." This phenomenon is called *semantic drift* [16].

A straightforward approach to avoid semantic drift is to terminate iterations before hitting generic patterns, but the optimal number of iterations is task-dependent and is hard to come by. The state-of-the-art bootstrapping algorithm *Espresso* [42] incorporates sophisticated scoring functions to cope with generic patterns, but Espresso still shows semantic drift unless iterations are terminated appropriately.

Previously proposed bootstrapping algorithms differ in how they deal with the problem of semantic drift. Also, there is an issue of generic patterns which deteriorates the quality of acquired instances. We will take the Espresso algorithm [42] as the example to explain common configuration for bootstrapping.

Another major drawback of bootstrapping is the lack of principled method for selecting optimal parameter values [36, 5]. It makes it hard to apply bootstrapping algorithms in practical settings.

In this work, we present a graph-theoretic analysis of Espresso-like bootstrapping algorithms. We argue that semantic drift is inherent in these algo-

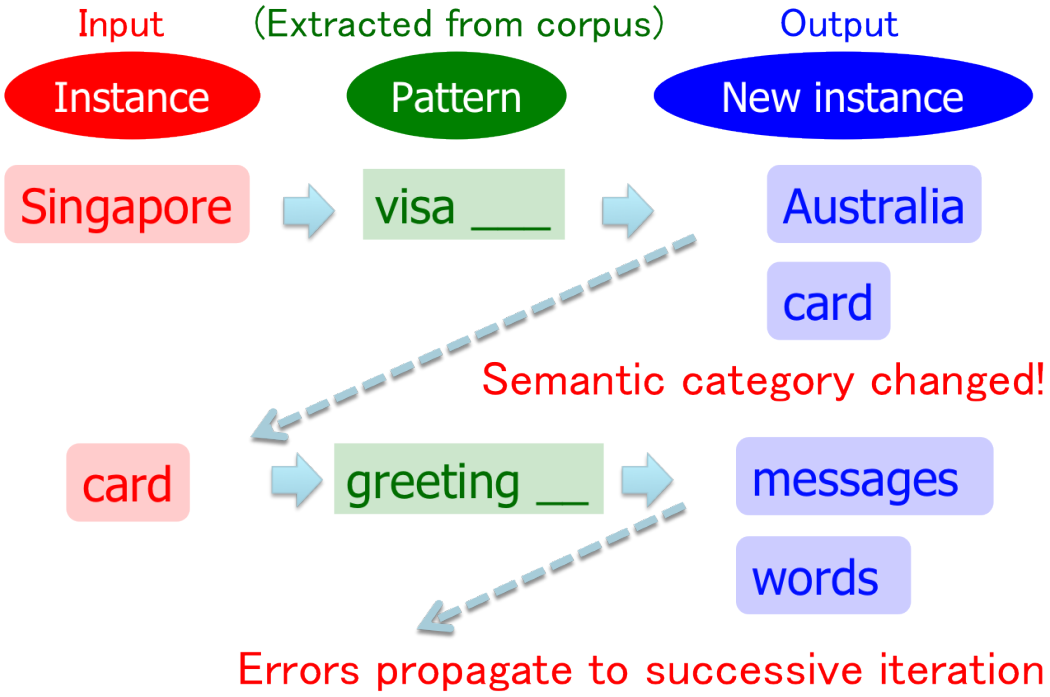

Figure 1.2. Semantic drift is the central problem of bootstrapping

rithms, and propose to use two graph-based algorithms that are theoretically less prone to semantic drift, as an alternative to bootstrapping.

The rest of this thesis is organized as follows. We overview the common setting of bootstrapping algorithms and compare various kinds of bootstrapping algorithms in Chapter 2. We will describe the Espresso algorithm in Chapter 3. We conducted an experiment to compare Espresso algorithm with other bootstrapping algorithms. In Chapter 4, we present a simplified version of Espresso to show the root of semantic drift, and apply two graph-based algorithms used in link analysis community to compute relatedness between instances. We then evaluate the algorithms in three NLP tasks: the results are reported in Chapters 5, 6 and 7.

2

Overview of Bootstrapping Algorithms

Bootstrapping (or self-training) is a general framework for reducing the requirement of manual annotation. The idea of learning with a bootstrapping method was adopted for many tasks. The goal of these algorithms is to learn target instances, which are the words belonging to certain categories (e.g., cat for the Animal class), or in the case of relation extraction, the pairs of words standing in a particular relationship (e.g., pasta::food for is-a relationship), given the context patterns for the categories or relation types found in source data. We will first describe the common setting of bootstrapping algorithms, and then explain the difference of various kinds of bootstrapping algorithms.

The first step toward the acquisition of instances is to extract context patterns. In previous work, these are surface text patterns, e.g., X such as Y, for extracting words in an is-a relation. The extracted context patterns must then be assigned a score reflecting their usefulness in extracting the instances of a desired type. Frequency is a poor metric here, because frequent patterns may be extremely generic, appearing across multiple categories. Previously proposed methods differ in how to assign the desirability scores to the patterns they find and in using the score to extract instances, as well as in the treatment of generic patterns, whose precision is low but whose recall is high.

Espresso and other bootstrapping methods [61, 1, 42, 16] alternate two phases: *pattern extraction* and *instance extraction*. For example, suppose we would like to learn hyponym (X is-a Y) relations. In pattern extraction phase, patterns like "Y such as X" which co-occur frequently with a seed such as (cat, animal) will be selected from a corpus. In instance extraction phase, on the other hand, new instances like (sparrow, bird) which co-occur with the patterns will be acquired and used for the next iteration.

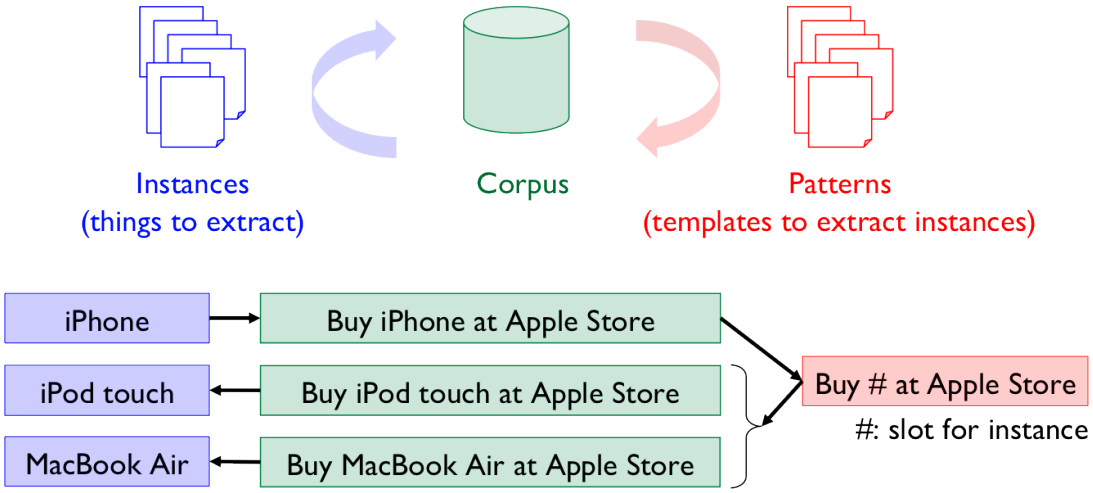

Figure 2.1. Two phases of bootstrapping: pattern and instance extraction.

We describe these phases below, along with the parameters that control each phase.

Phase 1. Pattern Extraction Induce patterns from a corpus given seed instances. Patterns may be surface text patterns, lexico-syntactic patterns, and/or just features. Confidence score is assigned to each pattern depending on co-occurrence strength to seed instances. Only top highest k patterns are selected (We call this process *pattern extraction*). It is necessary to assign low scores to generic patterns and high scores to patterns with high relatedness to the seed instances.

Phase 2. Instance Extraction Enumerate instances that co-occur with the patterns extracted in the pattern extraction phase. Compute confidence scores of enumerated instances and select high-confidence instances to add to the seed instance set. This process is called *instance extraction*. It is desirable to keep only high-confidence instances at this phase, as they are used as seed instances for the next iteration. Bootstrapping algorithms differ in the way to achieve this goal, i.e. whether (a) to output learned instances as *acquired instances* on each iteration to retain highly relevant instances learned in early iterations or (b) to output high-confidence instances on the last iteration instead of learning instances on each round.

Bootstrapping algorithms have parameters (i) *pattern reliability scoring metrics* and (ii) *the number of patterns to use for extraction of instances* for pattern

extraction phase; and (iii) *instance reliability scoring metrics*, (iv) *the number of instances to pass to the next iteration*. Also (v) *stopping criterion* is another parameter to control when to stop iterations.

Bootstrapping iterates the above two phases several times until stopping criteria are met. Acquired instances tend to become noisy as the iteration proceeds, so it is important to terminate before semantic drift occurs.

One of the major deficiencies in bootstrapping algorithms is that there is no principled manner to select optimal parameters and configurations [36]. How to cope with the problem of semantic drift differs from methods to methods, and it depends on tasks and domains.

2.1. Hearst (1992) Algorithm

Hearst (1992) [23] described a bootstrapping procedure for extracting words in hyponym (X *is-a* Y) relation, starting with manually given lexico-syntactic patterns. For example, a pattern "Y such as X" as in "... works by such authors as Herrick, Goldsmith, and Shakespeare." indicates that "Herick," "Goldsmith" and "Shakespeare" are hyponyms of "author." She then uses these terms to find new patters like "Y including X" as in "All common-law countries, including Canada and England ..." and repeats the process.

2.2. Yarowsky (1995) Algorithm

Yarowsky (1995) [61] presented an unsupervised word sense disambiguation (WSD) system which rivals supervised techniques. He exploits the "one sense per collocation" and "one sense per discourse" properties of natural language and incrementally learns a decision list with a few seed instances. His algorithm augments sense tagged data by iteratively training an intermediate classifier on classified data, using the resulting classifier to classify unlabeled data, and keeping the most confidently tagged instances for classified data to use on next iteration.

Although Yarowsky's original paper lacks theoretical justification of his algorithm, Abney [1] presented a thorough discussion on the Yarowsky algorithm. He extended the original Yarowsky algorithm to a new family of bootstrapping algorithms that are mathematically well understood.

2.3. Collins and Singer (1999) Algorithm

Collins and Singer (1999) [14] presented a new algorithm which combines Yarowsky's algorithm and co-training [8]. They induced a decision list for named entity classification starting from a few seed instances, by iteratively learning a spelling decision list and a decision list of contextual rules. They also presented a more general framework than the decision-list learning algorithm.

2.4. The Basilisk Algorithm (Thelen and Riloff 2002)

Thelen and Riloff (2002) [56] presented a framework called Basilisk, which extracts semantic lexicons for multiple categories. It starts with a small set of seed words and finds all patterns that match these seed words in the corpus. The bootstrapping process begins by selecting a subset of the patterns by the $R \log F$ metric [46]:

$$R \log F(pattern_i) = \frac{F_i}{N_i} \log_2(F_i) \tag{2.1}$$

where F_i is the number of category members extracted by $pattern_i$ and N_i is the total number of instances extracted by $pattern_i$. It then identifies instances by these patterns and scores each instance by the following formula:

$$AvgLog(word_i) = \frac{\sum_{j=1}^{P_i} \log_2(F_j + 1)}{P_i} \tag{2.2}$$

where P_i is the number of patterns that extract $word_i$. They used the average logarithm to select instances to balance the recall and precision of generic patterns. They added five best instances to the lexicon according to this formula, and the bootstrapping process starts again. Instances are cumulatively collected across iterations, while patterns are discarded at the end of each iteration.

2.5. Bilingual Bootstrapping (Li and Li 2004)

Li and Li (2004) [31] proposed a method called *Bilingual Bootstrapping*. It makes use of a translation dictionary and a comparable corpus to help disambiguate word senses in the source language, by exploiting the asymmetric many-to-many sense mapping relationship between words in two languages.

2.6. Mutual Exclusion Bootstrapping (Curran et al. 2007)

Curran et al. (2007) [16] presented an algorithm called *Mutual Exclusion Bootstrapping*, which minimizes semantic drift using mutual exclusion between semantic classes of learned instances. They prepared a list of so-called *stop classes* similar to a stop word list used in information retrieval to help bound the semantic classes. Stop classes are sets of terms known to cause semantic drift in particular semantic classes. However, stop classes vary from task to task and domain to domain, and human intervention is essential to create an effective list of stop classes.

2.7. Sekine and Suzuki (2007) Algorithm

For the purpose of choosing the set of context patterns that best characterizes the categories, Sekine and Suzuki (2007) [51] reported that none of the conventional co-occurrence metrics such as tf.idf, mutual information and chi-squared tests achieved good results on their task, and proposed a new measure, which is based on the number of different instances of the category a context c co-occurs with, normalized by its token frequency for all categories:

$$Score(c) = f_{type} \log_2 \frac{g(c)}{C} \tag{2.3}$$

$$g(c) = f_{type}(c) / F_{inst}(c) \tag{2.4}$$

$$C = \frac{f_{type}(ctop1000)}{F_{inst}(ctop1000)} \tag{2.5}$$

where f_{type} is the type frequency of instance terms that c co-occurs with in the category, F_{inst} is the token frequency of context c in the entire data and $ctop1000$ is the 1000 most frequent contexts. Since they started with a large and reliable named entity dictionary, and therefore used several hundred seed terms, they simply took the top-k highest-scoring contexts and extracted new named entities once and for all, without iteration. Generic patterns receive low scores, and are therefore ignored by this algorithm.

3 Espresso-style Bootstrapping Algorithms

In this chapter, we propose a new bootstrapping algorithm, called Tchai. We first discuss the Espresso framework [42] in some detail because Tchai is based on it. It is a general-purpose, minimally supervised bootstrapping algorithm that takes a few seed instances as input and iteratively learns surface patterns to extract more instances. The key to Espresso lies in its use of generic patterns: it assumes that correct instances captured by a generic pattern will also be instantiated by some reliable patterns, which mean high precision and low recall patterns. We then explain three modifications to the Espresso algorithm to further improve precision. We name the new Espresso-style algorithm *Tchai*. We experimentally show that the Espresso algorithm outperforms a bootstrapping algorithm called *Basilisk* described in the previous chapter. Thanks to the performance improvements from the modifications, the Tchai algorithm is more scalable than the Espresso algorithm. The Tchai algorithm achieves constantly higher precision than the Espresso algorithm, while keeping comparable recall.

3.1. The Espresso Algorithm

Pantel and Pennacchiotti [42] proposed a bootstrapping algorithm called Espresso to learn binary semantic relations such as *is-a* and *part-of* from a corpus. What distinguishes Espresso from other bootstrapping algorithms is that it benefits from generic patterns by using a principled measure of instance and pattern reliability. Previous methods use several tens of reliable patterns for instance extraction, excluding generic patterns to prevent semantic drift. However,

filtering generic patterns may harm coverage of the instance extraction and result in dropping recall. Espresso, on the other hand, defines sophisticated scoring functions to reduce the effect of semantic drift and compute a reliability score of an instance from all the patterns with small weight. This procedure improves recall and drastically increases the number of overall acquired instances.

Espresso starts from a small set of seed instances of a binary relation, finds a set of surface patterns P, selects the top-k patterns, extracts the highest scoring m instances, and repeats the process. Espresso ranks all patterns in P according to reliability r_π, and retains the top-k patterns for instance extraction. The value of k is increased by one after each iteration.

The key idea of Espresso is recursive definition of pattern-instance scoring metrics. They use pointwise mutual information (*pmi*) and define the reliability of a pattern p as its average strength of association across each input instance i in the set of instances I, weighted by the reliability of each instance i. The reliability scores of pattern p and instance i, denoted respectively as $r_\pi(p)$ and $r_\iota(i)$, are given as follows:

$$r_\pi(p) = \frac{\sum_{i \in I} \frac{pmi(i,p)}{\max\ pmi} r_\iota(i)}{|I|} \tag{3.1}$$

$$r_\iota(i) = \frac{\sum_{p \in P} \frac{pmi(i,p)}{\max\ pmi} r_\pi(p)}{|P|} \tag{3.2}$$

where P and I are sets of patterns and instances, and $|P|$ and $|I|$ are the numbers of patterns and instances, respectively. The *pmi* between instance $i = \{x, y\}$ and pattern p is estimated by:

$$pmi(i, p) = \log \frac{|x, p, y|}{|x, *, y||*, p, *|} \tag{3.3}$$

where $|x, p, y|$ is the frequency of pattern p instantiated with terms x and y (recall that Espresso is targeted at extracting binary relations) and where the asterisk represents a wildcard. They multiplied $pmi(i, p)$ with the discounting factor suggested in [43] to alleviate a bias towards infrequent events. In addition, max *pmi* is a maximum value of the pointwise mutual information over all instances and patterns.

The intuition behind these definitions is that a reliable pattern co-occurs with many reliable instances, and a reliable instance co-occurs with many reliable patterns.

Espresso uses Equations (3.1) for a pattern scoring function and (3.2) for an instance scoring function, respectively (see Chapter 2 for the meaning of these parameters), whereas other parameters rely on the tasks and need calibration. Unlike other bootstrapping algorithms, Espresso not only uses top-*k* ranked instances and patterns, but also all the instances and patterns with their confidence scores as weights, and thus it drastically improves recall while keeping precision high.

3.2. The Tchai Algorithm

In this section, we describe the modifications we made to Espresso to derive a new Espresso-style algorithm called Tchai, dedicated to learn semantic categories from web search query logs.

3.2.1 Segmentation of Query Logs

Bootstrapping methods often use some heuristics for finding the pattern boundaries in text. As we use query logs as the source of knowledge, we simply use everything but the instance string in a query as the pattern for the instance, in a manner similar to [41]. For example, the seed word JAL in the query "JAL+flight_schedule" yields the pattern "#+flight_schedule" [1] Note that we perform no word segmentation or boundary detection heuristics in identifying these patterns, which makes our approach fast and robust, as the segmentation errors introduce noise in extracted patterns, especially when the source data contains many out of vocabulary items.

3.2.2 Filtering Ambiguous Instances and Patterns

As mentioned above, the treatment of high-recall, low-precision generic patterns (e.g., #+map, #+animation) present a challenge to minimally supervised learning algorithms due to their ambiguity. In the case of semantic category acquisition, the problem of ambiguity is exacerbated, because not only the acquired patterns, but also the instances can be highly ambiguous. For example, once we learn an ambiguous instance such as Pokemon, it will start collecting

[1] ♯ indicates where the instance occurs in the query string, and + indicates a white space in the original Japanese query. The underscore symbol (_) means there was originally no white space; it is used merely to make the translation in English more readable.

patterns for multiple categories (e.g., Game, Animation and Movie), which is not desirable.

In order to control the negative effect of the generic patterns, Espresso introduces a confidence metric, which is similar to but different from the reliability score, and uses it to filter out the generic patterns falling below a confidence threshold. In our experiments, however, this metric did not produce a score that was substantially different from the reliability score. Therefore, we did not use a confidence metric, and instead opted for not filtering ambiguous instances and patterns, where we define ambiguous instance as one that induces more than 1.5 times the number of patterns of previously accepted reliable instances, and ambiguous (or generic) pattern as one that extracts more than twice the number of instances of previously accepted reliable patterns. As we will see in Section 3.3.5, this modification improves the precision of the extracted instances, especially in the early stages of iteration.

3.2.3 Scaling Factor in Reliability Scores

We made another modification to the Espresso algorithm to reduce the power of generic patterns. Espresso normalizes max *pmi* in terms of either instances or patterns instead of all instances and patterns as follows.

$$r_\iota(i) = \frac{\sum_{p \in P} \frac{pmi(i,p)}{\max_{p \in P} pmi} r_\pi(p)}{|P|} \tag{3.4}$$

$$r_\pi(p) = \frac{\sum_{i \in I} \frac{pmi(i,p)}{\max_{i \in I} pmi} r_\iota(i)}{|I|} \tag{3.5}$$

Since *pmi* ranges $[-\infty, +\infty]$, the point of dividing $pmi(i,p)$ by max *pmi* in Espresso is to normalize the reliability to $[-1, 1]$. However, using *pmi* directly to estimate the reliability of a pattern when calculating the reliability of an instance may lead to unexpected results because the absolute value of *pmi* is highly variable across instances and patterns. We define the *local* max *pmi* of the reliability of an instance to be the absolute value of the maximum *pmi* for a given instance, as opposed to taking the maximum for all instances in a given iteration. *Local* max *pmi* of the reliability of a pattern is defined in the same way.

This modification to Tchai increases the reliability of middle-frequent instances and patterns from Espresso. Typical bootstrapping algorithms use

only few patterns on each iteration, and thus ambiguous patterns affect precision especially in the early stages. Therefore, it is preferable to learn middle-frequent instances rather than infrequent but specific instances in order to find patterns related to the target category (these patterns cover middle frequent instances similar to the seed instances). As we will see in the next section, this modification has a large impact on the effectiveness of our algorithm.

Here, we modify Equation (3.3) to deal with unary pattern instead of binary relation. Thus,

$$pmi(i, p) = \log_2 \frac{|i, p|}{|i, *||*, p|} \tag{3.6}$$

is pointwise mutual information between i and p[2], $|i, *|$ and $|*, p|$ are the frequencies of pattern p and instance i in a given corpus, respectively, and $|i, p|$ is the frequency of pattern p which co-occurs with instance i.

3.2.4 Performance Improvements

Tchai, unlike Espresso, does not perform the pattern induction step between iterations; rather, it simply recomputes the reliability of the patterns induced at the beginning. Our assumption is that fairly reliable patterns will occur with at least one of the seed instances if they occur frequently enough in query logs. Since pattern induction is computationally expensive, this modification reduces the computation time by a factor of 400.

3.3. Experiment

In this section, we present an empirical comparison of Tchai with other systems such as Espresso and Basilisk [56]. These algorithms are summarized in Table 3.1.

[2]Although pointwise mutual information is usually defined as $\log_2 \frac{P(i,p)}{P(i)P(p)}$, Pantel and Pennacchiotti [42] have to approximate it using above equation because the true corpus size is not known (they used search engine hit counts from Google). However, we computed pointwise mutual information as usual, because we were able to estimate our corpus size correctly.

3.3.1 Introduction

Our method is based on the Espresso algorithm [42] for extracting binary lexical relations, improving it to work well on learning unary relations from query logs. The use of query data as a source of knowledge extraction offers some unique advantages over using regular text.

- Web search queries capture the interest of search users directly, while the distribution of the Web documents do not necessarily reflect the distribution of what people search [52]. The word categories acquired from query logs are thus expected to be more useful for the tasks related to search.
- Though user-generated queries are often very short, the words that appear in queries are generally highly relevant for the purpose of word classification.
- Many search queries consist of keywords, which means that the queries include word segmentation specified by users. This is a great source of knowledge for learning word boundaries for those languages whose regularly written text does not indicate word boundaries, such as Chinese and Japanese.

Although our work naturally fits into the larger goal of building knowledge bases automatically from text, to our knowledge we are the first to explore the use of Japanese query logs for the purpose of minimally supervised semantic category acquisition. Our work is similar to [51], whose goal is to augment a manually created dictionary of named entities by finding contextual patterns from English query logs. Our work is different in that it does not require a full-scale list of categorized named entities but a small number of seed words, and iterates over the data to extract more patterns and instances. Recent work by [38, 39] also uses English query logs to extract lexical knowledge, but their focus is on learning attributes for named entities, a different focus from ours.

3.3.2 Experimental Setup

Query logs: The data source for instance extraction is an anonymized collection of query logs submitted to Live Search from January to February 2007, taking the top 1 million unique queries. Queries with garbage characters are

Table 3.1. Algorithms for relation extraction

	# of seeds	target	corpus	language
Sekine & Suzuki	up to 600	categorized NE	query log	English
Basilisk	10	semantic lexicon	MUC-4 [54]	English
Espresso	up to 10	semantic relation	TREC-9 [58]	English
Tchai	5	semantic categories	query log	Japanese

removed. Almost all queries are in Japanese, and are accompanied by their frequency within the logs.

Target categories: Our task is to learn word categories that closely reflect the interest of web search users. We believe that a useful categorization of words is task-specific, therefore we did not start with any externally available ontology, but chose to start with a small number of seed words. For our task, we were given a list of 23 categories relevant for web search, with a manual classification of the 10,000 most frequent search words in the log of December 2006 (which we henceforth refer to as the 10K list) into one of these categories.[3] For evaluation, we chose two of the categories, Travel and Financial Services: Travel is the largest category containing 712 words of the 10K list (as all the location names are classified into this category), while Financial Services was the smallest, containing 240 words.

Systems: We compared three different systems that implement an iterative algorithm for lexical learning. They are briefly summarized in Table 3.1.

Basilisk The algorithm by Thelen and Riloff (2002) [56]

Espresso The algorithm by Pantel and Pennacchiotti (2006) [42]

Tchai The proposed algorithm described in this chapter.

For each system, we gave the same seed instances. The seed instances are the 5 most frequent words belonging to these categories in the 10K list; they

[3]The manual classification assigns only one category per word, which is not optimal given how ambiguous the category memberships are. However, it is also very difficult to reliably perform a multi-class categorization by hand.

Table 3.2. Seed instances for each category

Category	Seeds
Travel	jal, ana, jr, じゃらん (jalan), his
Finance	みずほ銀行 (Mizuho Bank), 三井住友銀行 (Sumitomo Mitsui Bank Corporation), jcb, 新生銀行 (Shinsei Bank), 野村證券 (Nomura Securities)

Table 3.3. Comparison with manual classification (10K list):Travel Category

		Manual classification		Not in the 10K list
		Travel	Not Travel	
Tchai	Travel	280	17	251
	Not Travel	0	7	125

Table 3.4. Comparison with manual classification (10K list): Finance Category

		Manual classification		Not in the 10K list
		Finance	Not Finance	
Tchai	Finance	41	30	30
	Not Finance	0	5	99

are given in Table 3.2. For the Travel category, "jal" and "ana" are airline companies, "jr" stands for Japan Railways, "jalan" is an online travel information site, and "his" is a travel agency. In the Finance category, three of them are banks, and the other two are a securities company and a credit card firm. Basilisk starts by extracting 20 patterns, and adds 100 instances per iteration. Espresso and Tchai start by extracting 5 patterns and add 200 instances per iteration. Basilisk and Tchai iterated 20 times, while Espresso iterated only 5 times due to computation time.

3.3.3 Results of the Tchai Algorithm

Tables 3.3 and 3.4 present the results of the Tchai algorithm compared to the manual classification. The figures "in the 10K list" show precision of the algorithm; whereas the figures "not in the 10K list" show whether the algorithm finds infrequent instances in web search query logs. Table 3.3 shows the results for the Travel category. The precision of Tchai is very high: out of the

Table 3.5. Sample of extracted instances

Type	Instance
Location	トルコ (Turkey), ラスベガス (Las Vegas), バリ島 (Bali Island)
Travel agencies	jtb, トクー (http://www.tocoo.jp/), yahoo (Yahoo! Travel), net cruiser
Attractions	ディズニーランド (Disneyland), usj (Universal Studio Japan)
Hotels	帝国ホテル (Imperial Hotel), リッツ (Ritz)
Transportation	京浜急行 (Keihin Express), 奈良交通 (Nara Kotsu Bus Lines)

297 words classified into the Travel category that were also in the 10K list, 280 (92.1%) were learned correctly. As the 10K list contained 712 words in the Travel category, the recall against that list is fairly low (ca.40%). The primary reason for this is that all location names are classified as Travel in the 10K list, and 20 iterations are not enough to enumerate all frequent location names. Another reason is that the 10K list consists of queries but our algorithm extracts instances. This sometimes causes a mismatch, e.g.,Tchai extracts "Ritz" but the 10K list contains "Ritz Hotel".

It turned out that the 17 instances that represent the precision error were due to the ambiguity of hand labeling, as in "Tokyo Disneyland" which is a popular travel destination, but is classified as Entertainment in the manual annotation. We were also able to correctly learn 251 words that were not in the 10K list according to manual verification; we also harvested 125 new words incorrectly into the Travel category, but these words include common nouns related to Travel, such as "fishing" and "rental car".

On the other hand, Table 3.4 shows the results for the Finance category. It exhibits a similar trend with the Travel category, but fewer instances are extracted. This is because the Finance category is a closed category which does not have many named entities such as location names. It is one of the future work to mine infrequent instances from query logs.

Sample instances harvested by our algorithm are given in Table 3.5. It includes subclasses of travel-related terms, for some of which no seed words were given (such as Hotels and Attractions). We also note that segmentation errors are entirely absent from the collected terms, demonstrating that query logs are in fact excellently suited for acquiring new words for languages with no explicit word segmentation in text.

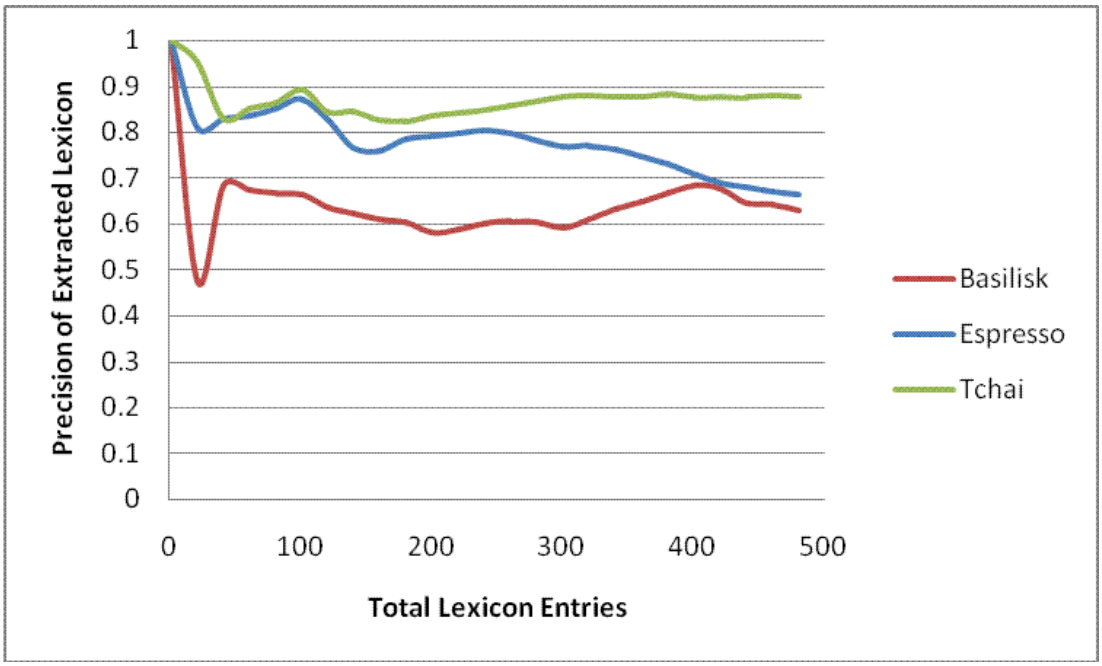

Figure 3.1. Precision for each system: Travel

3.3.4 Comparison with Basilisk and Espresso

Figures 3.1 and 3.2 show the precision results comparing Tchai with Basilisk and Espresso for the Travel and Finance categories. Tchai outperforms Basilisk and Espresso for both categories: its precision is constantly higher for the Travel category, and it achieves excellent precision for the Finance category, especially in early iterations. The differences in behavior between these two categories are due to the inherent size of these categories. For the smaller Finance category, Basilisk and Espresso both suffered from the effect of generic patterns such as "#ホームページ" (homepage) and "#カード" (card) in early iterations, whereas Tchai did not select these patterns. The more patterns used for instance extraction, the more likely bootstrapping algorithms tend to select generic patterns. Thus, it is necessary to investigate a way to calibrate parameters for each algorithm and category.

Comparing these algorithms in terms of recall is more difficult, as the complete set of words for each category is not known. However, we can estimate the relative recall given the recall of another system. Pantel and

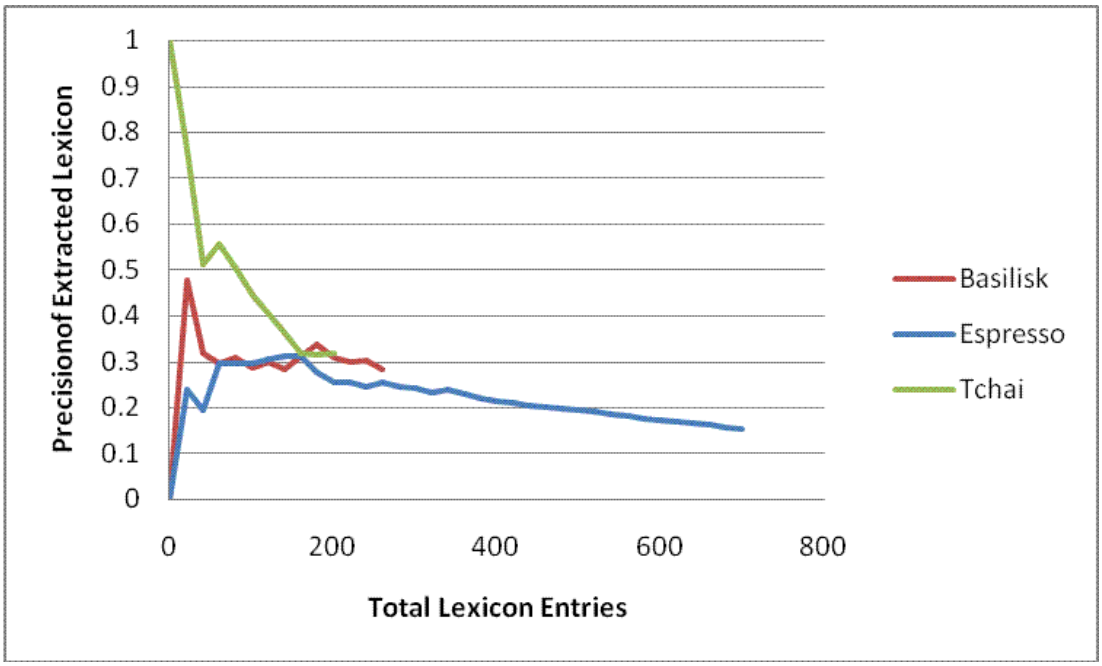

Figure 3.2. Precision for each system: Finance

Ravichandran (2004) [43] defined relative recall as:

$$R_{A|B} = \frac{R_A}{R_B} = \frac{C_A/C}{C_B/C} = \frac{C_A}{C_B} = \frac{P_A \times |A|}{P_B \times |B|} \tag{3.7}$$

where $R_{A|B}$ is the relative recall of system A given system B, C_A and C_B are the number of correct instances of each system, and C is the number of true correct instances. C_A and C_B can be calculated by using the precision, P_A and P_B, and the number of instances from each system. Using this formula, we estimated the relative recall of each system relative to Espresso.

Tables 3.6 and 3.7 show that Tchai achieved the best results in both precision and relative recall in the Travel category. In the Finance category, Espresso received the highest relative call but the lowest precision. This is because Tchai uses a filtering method so as not to select generic patterns and instances. The use of generic patterns did not improve as much as [42] reported even in the Finance category, therefore it is safe to say that our simple filtering of the Tchai algorithm is enough to filter generic patterns.

Table 3.8 shows the context patterns acquired by different systems after 4 iterations for the Travel category.[4]The patterns extracted by Basilisk are not

[4]Note that Basilisk and Espresso use context patterns only for the sake of collecting in-

Table 3.6. Relative recall: Travel

Systems	# of instances	Precision	Relative recall
Basilisk	651	63.4%	1.26
Espresso	500	65.6%	1.00
Tchai	680	80.6%	1.67

Table 3.7. Relative recall: Finance

Systems	# of instances	Precision	Relative recall
Basilisk	278	27.3%	0.70
Espresso	704	15.2%	1.00
Tchai	223	35.0%	0.73

entirely characteristic of the Travel category. For example, "p#sonic" and "google+#lytics" only match the seed word "ana", and are clearly irrelevant to the category. Basilisk uses token count to estimate the score of a pattern, which may explain the extraction of these patterns. Both Basilisk and Espresso identify location names as context patterns (e.g., "#東京" (Tokyo), "#九州" (Kyushu)), which may be too generic to be characteristic of the category. In contrast, Tchai finds context patterns that are highly characteristic, including terms related to transportation ("#+格安航空券" (discount plane ticket), "#マイレージ" (mileage)) and accommodation ("#+ホテル" (hotel)).

3.3.5 Contributions of Tchai Components

In this section, we examine the contribution of each modification to the Espresso algorithm we made in Tchai.

Figure 3.3 compares the original Espresso algorithm and the modified Espresso algorithm which performs the pattern induction step only at the beginning of the bootstrapping process, as described in Section 3.2.4. Although there is no significant difference in precision between the two systems, this modification greatly improves the computation time and enables efficient extraction of instances. We believe that our choice of the seed instances to be the most frequent words in the category produces sufficient patterns for ex-

stances, and are not interested in the patterns per se. However, they can be quite useful in characterizing the semantic categories they are acquired for, so we chose to compare them here.

Table 3.8. Sample of extracted patterns for each algorithm

Systems	Pattern
Basilisk	♯東日本 (east Japan), ♯西日本 (west Japan), p♯sonic, ♯時刻表 (timetable), ♯九州 (Kyushu), ♯+マイレージ (mileage), ♯バス (bus), google+♯lytics, ♯+料金 (fee), ♯+国内 (domestic), ♯ホテル (hotel)
Espresso	♯バス, 日本♯ (Japan), ♯ホテル, ♯道路 (road), ♯イン (inn), フジ♯ (Fuji), ♯東京 (Tokyo), ♯料金, ♯九州, ♯時刻表, ♯+旅行 (travel), ♯+名古屋 (Nagoya)
Tchai	♯+ホテル, ♯+ツアー (tour), ♯+旅行, ♯予約 (reservation), ♯+航空券 (flight ticket), ♯+格安航空券 (discount flight ticket), ♯+マイレージ, 羽田空港+♯ (Haneda Airport)

tracting new instances.

Figure 3.4 illustrates the effect of each modification proposed in Sections 3.2.2 and 3.2.3 for the Tchai algorithm on the Travel category. Each line in the graph corresponds to the Tchai algorithm with and without the modification. It shows that the modification to the max *pmi* function (purple; –noscaling) contributes most significantly to the improved accuracy of our system. The filtering of generic patterns (green; –nofilter) does not show a large effect in the precision of the acquired instances for this category, but produces steadily better results than the system without it.

3.4. Discussion

We propose a minimally supervised bootstrapping algorithm called Tchai. The main contribution of this work is to adapt the general-purpose Espresso algorithm to work well on the task of learning semantic categories of words from query logs. The proposed method not only has a superior performance in the precision of the acquired words into semantic categories, but is faster and collects more meaningful context patterns for characterizing the categories than the unmodified Espresso algorithm. We have also shown that the proposed method requires no pre-segmentation of the source text for the purpose of knowledge acquisition.

Bootstrapping algorithms, however, often have many parameters to optimize and hence are difficult to use in practice. We will see in the next chapter

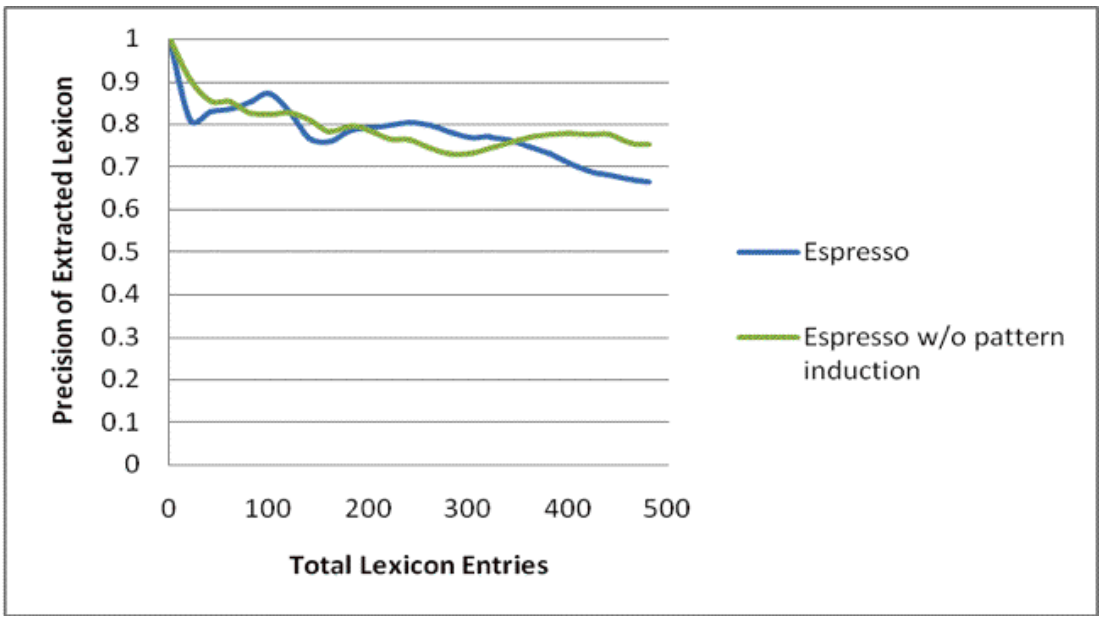

Figure 3.3. Effect of modification to pattern extraction step

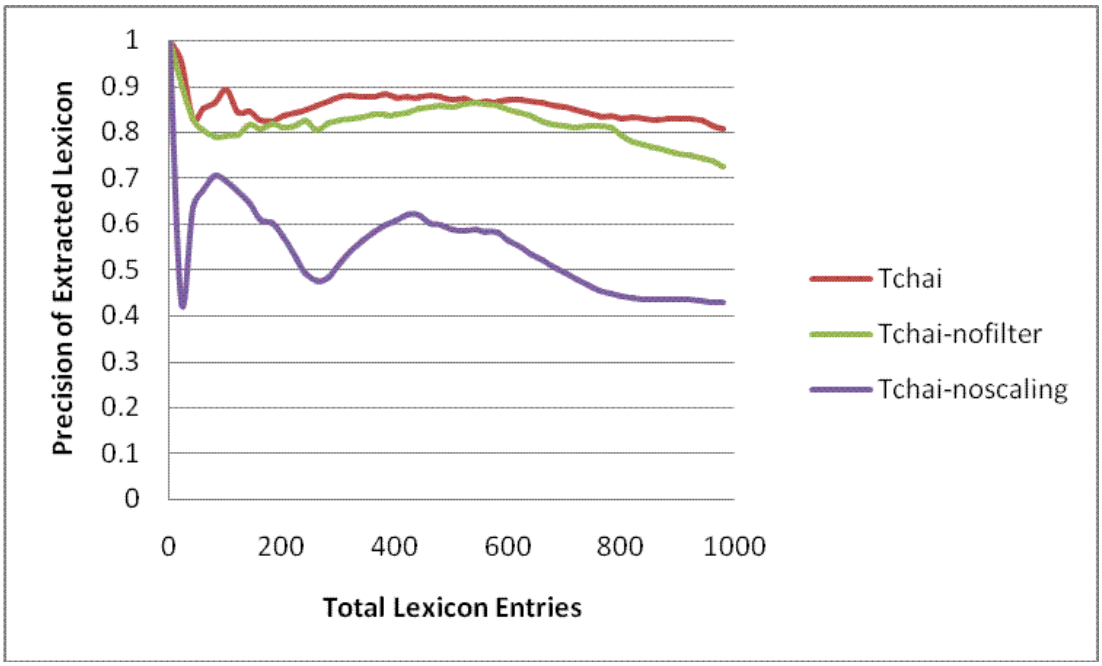

Figure 3.4. Precision of *Tchai* in Travel Category

that parameter tuning is crucial for Espresso-style bootstrapping algorithms to achieve the best performance. Also, we present a graph-based analysis of Espresso-style bootstrapping algorithms to elicit the cause of semantic drift of bootstrapping algorithms. We will see that semantic drift is persistent across NLP tasks such as semantic category acquisition and word sense disambiguation in Chapters 4, 5 and 7, and that proposed methods overcome the problem of semantic drift in Chapters 5, 6 and 7.

4

Graph-based Analysis of Espresso-style Bootstrapping Algorithms

A major drawback of bootstrapping is the lack of principled method for selecting optimal parameter values [36, 5]. Also, there is an issue of generic patterns which deteriorates the quality of acquired instances. Previously proposed bootstrapping algorithms differ in how they deal with the problem of semantic drift. We have taken recently proposed Espresso algorithm as the example to explain common configuration for bootstrapping.

It is known that bootstrapping often acquires instances not related to seed instances. For example, consider the task of collecting the names of common tourist sites from web corpora. Given words like "Geneva" and "Bali" as seed instances, bootstrapping would eventually learn *generic patterns* such as "pictures" and "photos," which also co-occur with many other unrelated instances. The subsequent iterations would likely acquire frequent words that co-occur with these generic patterns, such as "Britney Spears." This phenomenon is called *semantic drift* [16].

A straightforward approach to avoid semantic drift is to terminate iterations before hitting generic patterns, but the optimal number of iterations is task dependent and is hard to come by. The recently proposed *Espresso* [42] algorithm described in Chapter 3 incorporates sophisticated scoring functions to cope with generic patterns, but as we have pointed out in the previous chapter, Espresso still shows semantic drift unless iterations are terminated appropriately.

Another deficiency in bootstrapping is its sensitivity to many parameters such as the number of seed instances, the stopping criterion of iteration, the number of instances and patterns selected on each iteration, and so forth.

These parameters also need to be calibrated for each task.

In this chapter, we present a graph-theoretic analysis of Espresso-like bootstrapping algorithms. We argue that semantic drift is inherent in these algorithms, and apply two graph-based algorithms that are theoretically less prone to semantic drift, as an alternative to bootstrapping.

First, we analyze in Section 4.1 a bootstrapping algorithm (*Simplified Espresso*) which can be thought of as a degenerate version of Espresso. Simplified Espresso is simple enough to allow an algebraic treatment, and its equivalence to Kleinberg's HITS algorithm [27] is shown. An implication of this equivalence is that semantic drift in this bootstrapping algorithm is essentially the same phenomenon as topic drift observed in link analysis. Another implication is that semantic drift is inevitable in Simplified Espresso as it converges to the same score vector regardless of seed instances.

The original Espresso also suffers from the same problem as its simplified version does. It incorporates heuristics not present in Simplified Espresso to reduce semantic drift, but these heuristics have limited effect as we demonstrate in Section 4.1.3.

In Section 4.2, we propose two graph-based algorithms to reduce semantic drift. These algorithms are used in link analysis community to reduce the effect of topic drift. In Section 4.2.3, we show connection between label propagation approach and the proposed graph-based methods. In Chapter 5 we apply them to the task of word sense disambiguation (WSD) on Senseval-3 Lexical Sample Task and verify that they indeed reduce semantic drift. In Chapter 6 we explore the task of bilingual dictionary construction and semantic category acquisition to apply the proposed algorithms in Chapter 7.

4.1. Analysis of an Espresso-like Bootstrapping Algorithm

4.1.1 Simplified Espresso

Let us consider a simple bootstrapping algorithm illustrated in Figure 4.1, in order to elucidate the cause of semantic drift.

As before, let $|I|$ and $|P|$ be the numbers of instances and patterns, respectively. The algorithm takes a seed vector $\mathbf{i}_0$, and a pattern-instance co-occurrence matrix M as input. $\mathbf{i}_0$ is a $|I|$-dimensional vector with 1 at the position of seed instances, and 0 elsewhere. M is a $|P| \times |I|$-matrix whose

input: seed vector $\mathbf{i}_0$
input: pattern-instance co-occurrence matrix M
output: instance score vector $\mathbf{i}$
output: pattern score vector $\mathbf{p}$
1: $\mathbf{i} = \mathbf{i}_0$
2: **repeat**
3: $\quad \mathbf{p} \leftarrow M\mathbf{i}$
4: $\quad$ Normalize $\mathbf{p}$
5: $\quad \mathbf{i} \leftarrow M^T\mathbf{p}$
6: $\quad$ Normalize $\mathbf{i}$
7: **until** $\mathbf{i}$ and $\mathbf{p}$ have both converged
8: **return** $\mathbf{i}$ and $\mathbf{p}$

Figure 4.1. A simple bootstrapping algorithm

(p, i)-element $[M]_{pi}$ holds the (possibly re-weighted) number of co-occurrence of pattern p and instance i in the corpus. If both $\mathbf{i}$ and $\mathbf{p}$ have converged, the algorithm returns the pair of $\mathbf{i}$ and $\mathbf{p}$ as output.

This algorithm, though simple, can encode Espresso's update formulae (3.1) and (3.2) as Steps 3 through 6 if we pose

$$[M]_{pi} = \frac{pmi(i, p)}{\max pmi}, \tag{4.1}$$

and normalize $\mathbf{p}$ and $\mathbf{i}$ in Steps 4 and 6 by

$$\mathbf{p} \leftarrow \mathbf{p}/|I| \quad \text{and} \quad \mathbf{i} \leftarrow \mathbf{i}/|P|, \tag{4.2}$$

respectively.

This specific instance of the algorithm of Figure 4.1 obtained by specialization through Equations (4.1) and (4.2), will be henceforth referred to as *Simplified Espresso*. Indeed, it is an instance of the original Espresso in which the iteration is not terminated until convergence, all instances are carried over to the next iteration, and instances are not cumulatively learned.

4.1.2 Simplified Espresso as Link Analysis

Let n denote the number of times Steps 2–8 are iterated. Plugging (4.1) and (4.2) into Steps 3–6, we see that the score vector of instances after the nth

iteration is

$$\mathbf{i}_n = A^n \mathbf{i}_0 \tag{4.3}$$

where

$$A = \frac{1}{|I||P|} M^T M. \tag{4.4}$$

Suppose matrix A is irreducible; i.e., the graph induced by taking A as the adjacency matrix is connected. If n is increased and $\mathbf{i}_n$ is normalized on each iteration, $\mathbf{i}_n$ tends to the principal eigenvector of A. This implies that no matter what seed instances are input, the algorithm will end up with the same ranking of instances, if it is run until convergence. Because $A = \frac{M^T M}{|I||P|}$, the principal eigenvector of A is identical to the authority vector of HITS [27] algorithm run on the graph induced by M. [1] This similarity of Equations (3.1), (3.2) and HITS is not discussed in [42].

As a consequence of the above discussion, semantic drift in simplified Espresso seems to be inevitable as the iteration proceeds, since the principal eigenvector of A need not resemble seed vector $\mathbf{i}_0$. A similar phenomenon is reported for HITS and is known as *topic drift*, in which pages of the dominant topic are ranked high regardless of the given query. [6]

Unlike HITS and Simplified Espresso, however, Espresso and other bootstrapping algorithms [61, 47], incorporate heuristics so that only patterns and instances with high confidence score are carried over to the next iteration. We denote this heuristics as *filtering* and experiment the effect of the heuristics in the next section.

4.1.3 Convergence Process of Espresso

To investigate the effect of semantic drift on Espresso with and without the heuristics of selecting the most confident instances on each iteration (i.e., the original Espresso and Simplified Espresso of Section 4.1.2), we apply them to the task of word sense disambiguation.

The task of WSD is to correctly predict the senses of test instances whose true sense is hidden from the system, using training data and their true senses.

[1] As long as the relative magnitude of the components of vector $\mathbf{i}_n$ is preserved, the vector can be normalized in any way on each iteration. Hence HITS and Simplified Espresso both converge to the principal eigenvector of A, even though use different normalization.

To predict the sense of a given instance i, we apply k-nearest neighbour algorithm.

Given a test instance i, its sense is predicted with the following procedure:

1. Compute the instance-pattern matrix M from the entire set of instances. We defer the details of this step to Section 5.3.

2. Run Simplified Espresso and Espresso using the given instance i as the only seed instance.

3. After the termination of the algorithm, select k training instances with the highest scores in the score vector $\mathbf{i}$ output by the algorithm.

4. Since the selected k instances are training instances, their true senses are accessible. Choose the majority sense s from these k instances, and output s as the prediction for the given instance i. When there is a tie, output the sense of the instance with the highest score in $\mathbf{i}$.

Note that only Step (3) uses sense information. Steps (3) and (4) can be regarded as a k-nearest neighbour method [15] using the confidence score of instances of (Simplified) Espresso after termination of the algorithms as proximity measure. [2] k-nearest neighbour is simple but reported to achieve high performance in the task of word sense disambiguation. [35]

As we can see from Step (2), the instance to disambiguate is used as a seed instance. Therefore, unlike relation extraction and named entity classification tasks there is no choice in selecting a seed in word sense disambiguation task. It is desirable since we can exclude the effect of the choice of seeds from evaluation.

We use the Senseval-3 Lexical Sample (S3LS) Task data. [3] and the standard training-test split provided with the data set.

In this section we choose a word "bank" to see typical behaviour for each algorithm. We report overall results in Chapter 5. In S3LS dataset, there are 394 instances of word "bank" and their occurring context in this dataset, and each of them is annotated with its true sense. Of the ten senses of *bank*, the most frequent is the bank as in "bank of the river."

[2] We opted for a simplistic approach instead of using weighted voting or Support Vector Machines because our goal was not to build a state-of-the-art word sense disambiguation system.

[3] http://www.senseval.org/senseval3/data.html

On each iteration of Espresso, we cleared all but the 100 top-scoring instances in the instance vector and used only retained instances to compute the pattern score vector. The number of non-zeroed instance scores was increased by 100 on each iteration. On the other hand, we cleared all but the 20 top-scoring patterns in the pattern vector and used only retained patterns to compute the instance score vector on each iteration. These numbers are different from the original paper [42] because we conducted a preliminary experiment to find optimal values. However, the number of non-zeroed patterns was increased by 1 on each iteration just like [42] [4] The values of other parameters remain the same as those for simplified Espresso in Section 4.1.1. More specifically, we used Equations (3.1) and (3.2) as scoring functions and instances were learned according to the final scores.

Furthermore, although [42] shows a stopping criterion for the task of relation extraction, the criterion lacks generality and is not readily applicable to other tasks such as word sense disambiguation. Therefore, we did not explicitly set a stopping criterion; rather, we ran the algorithm until convergence and observed accuracy on each iteration.

Figure 4.2 shows the convergence process of Simplified-Espresso and Espresso. X-axis indicates the number of bootstrapping iterations and Y-axis indicates the accuracy ($= \frac{|instances\ being\ assigned\ the\ correct\ sense|}{|total\ attempts\ of\ prediction|}$).

Simplified Espresso tends to select the most frequent sense as the iteration proceeds, and after nine iterations it selects the most frequent sense ("the bank of the river") regardless of the seed instances. As expected from the discussion in Section 4.1.2, generic patterns gradually got more weight and semantic drift occurred in later iterations. Indeed, the ranking of the instances after convergence was identical to the HITS authority ranking computed from instance-pattern matrix M (i.e., the ranking induced by the dominant eigenvector of $M^T M$).

On the other hand, Espresso suffers less from semantic drift. The final recall achieved was 0.773 after convergence on the 20th iteration, outperforming the most-frequent sense baseline by 0.10. However, a closer look reveals that the filtering heuristics is limited in effectiveness.

Figure 4.3 plots the learning curve of Espresso on the set of test instances. We show recall ($\frac{|instances\ being\ assigned\ the\ correct\ sense|}{|total\ instances\ to\ be\ predicted|}$) of each sense to see how

[4] The number of initial patterns is relatively large because of a data sparseness problem in WSD, unlike relation extraction and named entity recognition. Also, WSD basically uses more features than relation extraction and thus it is hard to determine the stopping criterion based on the number and scores of patterns, as [42] does.

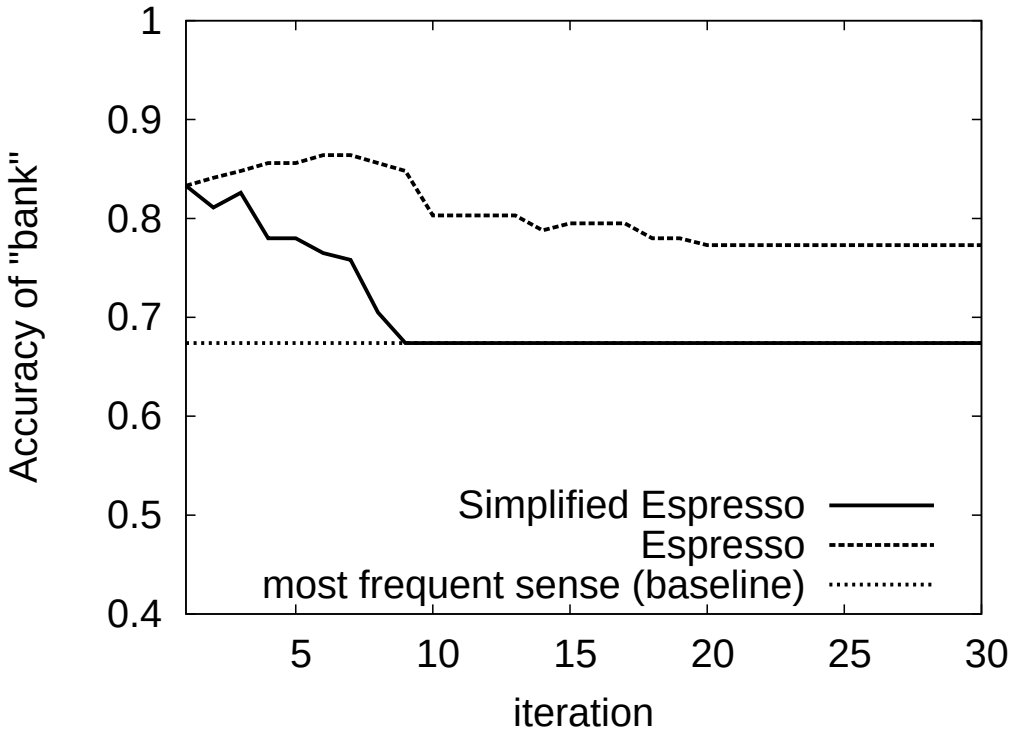

Figure 4.2. Accuracy of Simplified Espresso and Espresso

Espresso tends to select the most frequent sense. If semantic drift takes place, the number of instances predicted as the most frequent sense should increase as the iteration proceeds, resulting in increased recall on the most frequent sense and decreased recall on other senses. Figure 4.3 exactly exhibit this trend, meaning that Espresso is not completely free from semantic drift. Figure 4.2 also shows that the recall of Espresso starts to decay after the seventh iteration.

4.2. Two Graph-based Algorithms to Reduce Semantic Drift

We explore two graph-based methods which have the advantage of Espresso to harness the property of generic patterns by the mutual recursive definition of instance and pattern scores. They also have less parameters than bootstrapping, and are less prone to semantic drift.

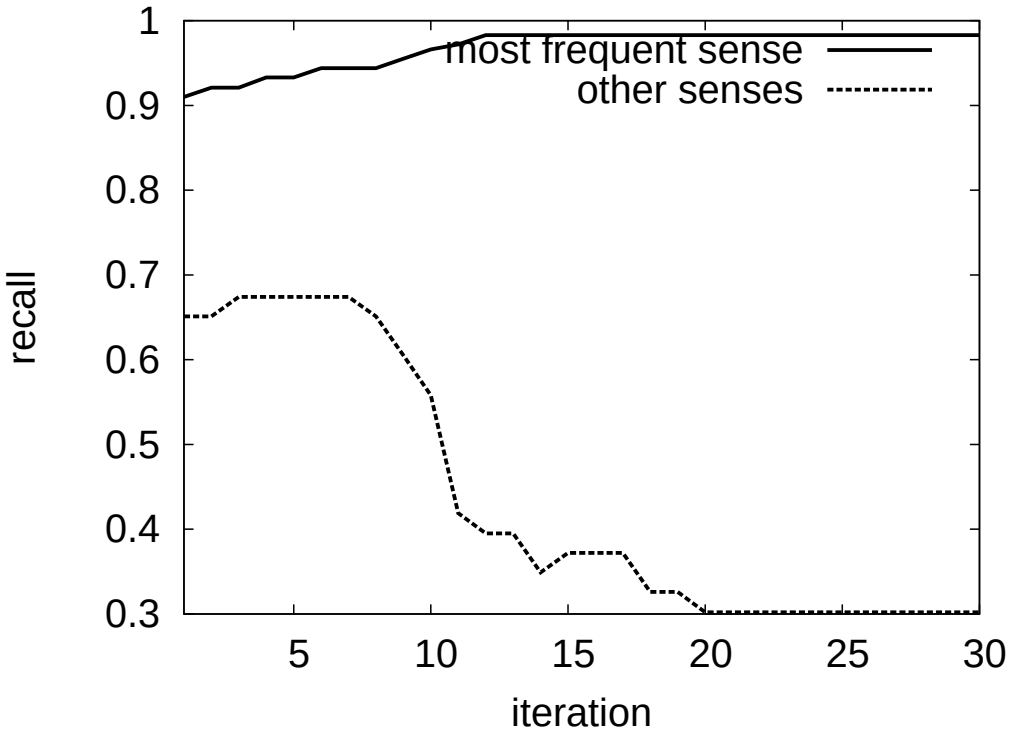

Figure 4.3. Recall of Espresso on the instances having "bank of the river" and other senses

4.2.1 Von Neumann Kernel

Kandola et al. [26] proposed the *von Neumann kernels* for measuring similarity of documents using words. If we apply the von Neumann kernels to the pattern-instance co-occurrence matrix instead of the document-word matrix, the relative importance of an instance to seed instances can be estimated.

Let $A = M^T M$ be the instance similarity matrix obtained from pattern-instance matrix M, and λ be the principal eigenvalue of A. The *von Neumann kernel matrix* K_α with diffusion factor α $(0 \leq \alpha < \lambda^{-1})$ is defined as follows:

$$K_\alpha \;=\; A \sum_{n=0}^{\infty} \alpha^n A^n = A(I - \alpha A)^{-1}. \tag{4.5}$$

The similarity between two instances i, j is given by the (i, j) element of K_α. Hence, the i-th column vector ($=$ i-th row vector, since K_α is symmetric) can be used as the score vector for seed instance i.

Ito et al. [24] showed that the von Neumann kernels represent a mixture of the co-citation relatedness and Kleinberg's HITS importance. They compute the weighted sum of all paths between two nodes in the co-citation graph induced by $A = M^T M$. The $(M^T M)^n$ term of smaller n corresponds to the relatedness to the seed instances, and the $(M^T M)^n$ term of larger n corresponds to HITS importance. The von Neumann kernels calculate the weighted sum of $(M^T M)^n$ from $n = 1$ to ∞, and therefore smaller diffusion factor α results in ranking by relatedness, and larger α returns ranking by HITS importance.

Given these definitions, we see that the $(M^T M)^n$ term of $n = 1$ corresponds to the co-citation relatedness. Also, $(M^T M)^n$ term of larger n tends to Kleinberg's HITS importance. Therefore, when diffusion factor is small the algorithm gives a ranking obeying relatedness whereas when diffusion factor is large it gives a ranking obeying HITS importance.

As a result, if diffusion factor α is configured properly, it is expected that resulting similarity measure will balance the most frequent sense and other rare senses. We verify this claim in Section 5.4.

4.2.2 Regularized Laplacian Kernel

The von Neumann kernels can be regarded as a mixture of relatedness and importance, and diffusion factor α controls the trade-off between relatedness and importance. Because there is only one parameter, to optimize diffusion factor is much easier than tweaking bootstrapping algorithms. In practice, however, setting the right parameter value becomes an issue. We solve this problem by the regularized Laplacian [53, 13] proposed in link analysis community as a relatedness measure. It has diffusion parameter just as von Neumann kernel does, but is stable across its diffusion factors and thus supposed to be easy to calibrate. In addition, unlike von Neumann kernel, it does not become an importance measure even though diffusion factor is large; instead, it remains a relatedness measure. This property is appealing to the tasks that bootstrapping algorithms are typically applied to. In fact, this kernel tends to a uniform matrix as $\alpha \to \infty$.

Let G be a weighted undirected graph whose adjacency (weight) matrix is a symmetric matrix A. The (normalized) graph Laplacian $\mathcal{L}$ of a graph G is defined as follows:

$$\mathcal{L} = I - D^{-1/2} A D^{-1/2} \tag{4.6}$$

where D is a diagonal matrix, and the ith diagonal element $[D]_{ii}$ is given by

$$[D]_{ii} = \sum_j [A]_{ij}. \tag{4.7}$$

Here, $[A]_{ij}$ stands for the (i, j) element of A. By replacing A with $-\mathcal{L}$ in Equation (4.5) and deleting the first A, we obtain a *regularized Laplacian kernel*.

$$R_\alpha = \sum_{n=0}^{\infty} \alpha^n (-\mathcal{L})^n = (I + \alpha\mathcal{L})^{-1} \tag{4.8}$$

Again, $\alpha (0 \leq \alpha < \lambda^{-1})$ is called the diffusion factor.

Both the regularized Laplacian and the von Neumann kernels include arguments of higher order and thus take higher order correlation between instances into account.

The main difference between the regularized Laplacian and the von Neumann kernel lies in the normalization of A by $D^{-1/2}AD^{-1/2}$. The weight of edges connecting to nodes (instances) with high degree (which means that the instances co-occur with generic patterns) will get lowered by this normalization, and therefore the effect of generic patterns will be suppressed.

4.2.3 Connection to Label Propagation

Graph-based semi-supervised methods such as label propagation are known to achieve high performance with only a few seeds and have the advantage of scalability.

Figure 4.4 illustrates the process of label propagation using a seed term "singapore" to learn the Travel domain.

This is a bipartite graph whose left-hand side nodes are terms (instances) and right-hand side nodes are patterns. The strength of lines indicates relatedness between each node. The darker a node, the more likely it belongs to the Travel domain. Starting from "singapore," the pattern "♯ airlines" [5] is strongly related to "singapore," and thus the label of "singapore" will be propagated to the pattern. On the other hand, the pattern "♯ map" is a neutral pattern which co-occurs with terms other than the Travel domain such as "google" and "yahoo." Since the term "china" shares two patterns, "♯ airlines" and "♯ map," with "singapore," the label of the seed term "singapore" propagates to "china." "China" will then be classified in the Travel domain.

[5] ♯ is the place into which an instance fits.

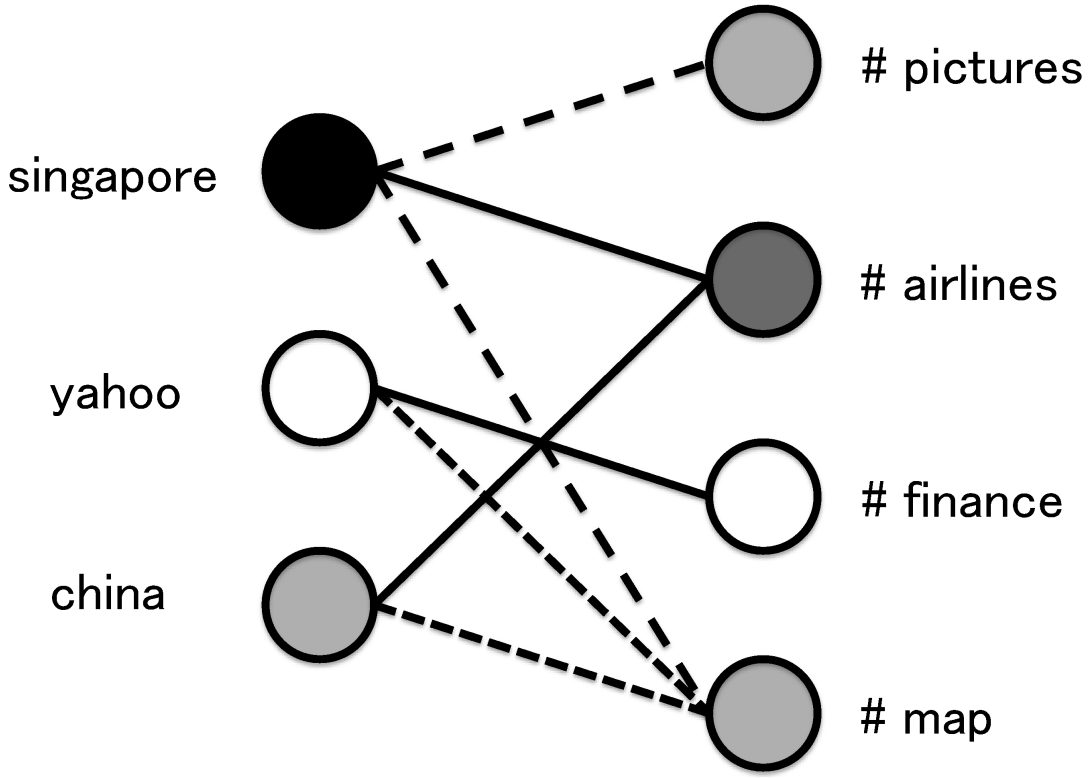

Figure 4.4. Labels of seeds are propagated to unlabeled nodes.

Figure 4.5 depicts a label propagation algorithm based on Zhou et al. (2004) [62]. Let $\mathcal{X}$ be a set of all instances. In the task of learning one category, $F(t)$ is a score vector of $\mathcal{X}$ on t-th iteration with dimension being the number of the element $\mathcal{X}$, $|\mathcal{X}|$. The value of the i-th dimension of $F(t)$ represents the degree that the i-th instance of $\mathcal{X}$, x_i belongs to the target category. Thus, $F(t)$ is a score vector of the target category. Given a seed instance set, if the set contains x_i, the score of the i-th element of the input, $F(0)$, holds 1, otherwise 0. The algorithm uses the instance similarity matrix A of dimension $|\mathcal{X}| \times |\mathcal{X}|$ to update the score vector; and finally outputs $F(t)$ after convergence.

In the task of learning two categories, $F(t)$ can be regarded as a vector of dimension $|\mathcal{X}|$. the score of the i-th element of $F(0)$ holds either 1 or -1 depending on the class, respectively.

In the task of learning more than $n(n \geq 3)$, $F(t)$ is a $|\mathcal{X}| \times n$ matrix. The j-th column vector corresponds to a score vector of a category j. To classify which class an instance belongs to, the instance is assigned the j-th class in

Input: Seed instance vector $F(0)$
Input: Instance similarity matrix A
Output: Instance score vector $F(t)$
1: Iterate $F(t+1) = \alpha AF(t) + (1-\alpha)F(0)$ until convergence

Figure 4.5. Simple label propagation algorithm

which it has the highest score in the j-th column vector. Note that this can be applied to the task of learning two categories as well, and the task of learning single category is the special case of this method with $n = 1$.

Label propagation has a parameter $\alpha (0 \leq \alpha < \lambda^{-1})$ where λ is the principal eigenvalue of A. Parameter α controls how much the labels of seeds are emphasized. As α approaches 0 it puts more weight on labeled instances, while as α increases it employs both labeled and unlabeled data.

However, this algorithm has a problem of semantic drift depending on the way to create a similarity matrix. Especially, when a similarity matrix is computed from an instance-pattern co-occurrence matrix as $A = W^T W$, the ranking of the output of the algorithm described in Figure 4.5 is identical to those of the authority vector of Kleinberg's HITS [27], and hence does not depend on seed instances [24]. In the previous section, we showed that this phenomenon is known as topic drift in HITS and is called as semantic drift in bootstrapping algorithms. Zhou et al. (2004) [62] set the diagonal element of a similarity matrix $A_{ii} = 0$ to solve the problem of semantic drift.

The difference between [62] and our work is that the way we create a similarity matrix and the application of the regularized Laplacian to the similarity matrix. The regularized Laplacian normalizes its weights when a node is connected to many other nodes. This normalization downweights generic patterns and thus makes the regularized Laplacian less prone to semantic drift.

In this way, label propagation gradually propagates the label of seed instances to neighbouring nodes, and optimal labels are given as the labels at which the label propagation process has converged.

Figure 4.6 describes label propagation based on the *regularized Laplacian*. Let a sample x_i be $x_i \in \mathcal{X}$, $F(0)$ be a score vector of x comprised of a label set $y_i \in \mathcal{Y}$, and $F(t)$ be a score vector of x after step t. *Instance-instance similarity matrix* A is defined as $A = W^T W$ where $W$is a row-normalized *instance-pattern matrix*. The (i, j)-th element of W_{ij} contains the normalized frequency of co-occurrence of instance x_i and pattern p_j. D is a diagonal degree matrix of N

Input: Seed instance vector $F(0)$
Input: Instance similarity matrix A
Output: Instance score vector $F(t)$

1: Construct the normalized Laplacian $\mathcal{L} = I - D^{-1/2}AD^{-1/2}$
2: Iterate $F(t+1) = \alpha(-\mathcal{L})F(t) + (1-\alpha)F(0)$ until convergence

Figure 4.6. The Laplacian label propagation algorithm

where the (i,i)-th element of D is given as $D_{ii} = \sum_j N_{ij}$. .

Graph-based minimally supervised methods such as label propagation make final classification using the scores of nodes assigned by weighted voting of labels on each iteration. We show that using the normalized Laplacian as a similarity matrix in label propagation gives the regularized Laplacian kernel [53] described in Section 4.2.

First, let us prove the sequence $F(t)$ converges to $F^* = (1-\alpha)(I+\alpha\mathcal{L})^{-1}F(0)$. By iterative algorithm,

$$F(t) = (\alpha(-\mathcal{L}))^{t-1}F(0) + (1-\alpha)\sum_{i=0}^{t-1}(\alpha(-\mathcal{L}))^i F(0).$$

Suppose $0 \leq \alpha \leq 1/2$. Since the eigenvalues of $(-\mathcal{L})$ lie in the range $[-2,0]$, we have

$$\lim_{t\to\infty}(\alpha(-\mathcal{L}))^{t-1} = 0$$

and

$$\lim_{t\to\infty}\sum_{i=0}^{t-1}(\alpha(-\mathcal{L}))^i = (I-\alpha(-\mathcal{L}))^{-1} = (I+\alpha\mathcal{L})^{-1}.$$

Therefore,

$$F^* = \lim_{t\to\infty} F(t) = (1-\alpha)(I+\alpha\mathcal{L})^{-1}F(0).$$

We can ignore $(1-\alpha)$ for classification, and hence get

$$F^* = (I+\alpha\mathcal{L})^{-1}F(0). \tag{4.9}$$

This is the same equation of the regularized Laplacian kernel [53] as we explained in Section 4.2, which means that the Laplacian label propagation virtually uses the regularized Laplacian kernel for weighted voting for the final classification.

However, the matrix inversion leads to $O(n^3)$ complexity, which is far from realistic in a real-world configuration. Nonetheless, it can be approximated by fixing the number of steps for label propagation. We will see how to calculate the regularized Laplacian efficiently in Chapter 7.

4.3. Discussion

In this chapter we propose a graph-based analysis of Espresso-style bootstrapping algorithms. We show that the source of semantic drift has the same root as topic drift, and present two graph-based algorithms as an alternative to bootstrapping. We will see in the following chapters the regularized Laplacian performs state-of-the-art for various natural language processing tasks. We have indicated connection between the regularized Laplacian and a variant of label propagation method. We will present an approximation to the regularized Laplacian and show that it scales to large amount of real-world data in Chapter 7.

5 Word Sense Disambiguation

Word sense disambiguation (WSD) is one of the central tasks of natural language processing. In recent years a number of shared tasks such as Senseval[1] and Semeval[2] have been held to evaluate WSD systems. However, most state-of-the-art WSD systems heavily rely on sense tagged corpora such as SemCor[3] and rich thesauri such as WordNet[4], which are not readily available to many languages. So far, reducing the cost of human annotation is one of the important problems for building WSD systems.

Yarowsky [61] first presented bootstrapping-based WSD system which rivals supervised techniques. The system exploits two properties of human language, that there is: (i) one sense per collocation, and (ii) one sense per discourse, and it iteratively learns classifiers to disambiguate polysemous words. Bootstrapping methods require only a small amount of instances to start with, but can easily multiply the number of labeled instances with minimal human annotation cost. There are many WSD systems using semi-supervised techniques [33, 31, 45, 37].

However, bootstrapping-based WSD is not free from semantic drift, as we described in Section 4.1.1. For example, suppose one would like to disambiguate a word "interest" with local bag-of-words features. Given seed instance with the "a feeling of wanting to know or learn about" sense, bootstrapping would eventually learn generic patterns such as your, rate in "rate your *interest*," which also co-occur with many other unrelated instances like

[1] http://www.senseval.org/

[2] http://nlp.cs.swarthmore.edu/semeval/

[3] http://www.cs.unt.edu/ rada/downloads.html#semcor

[4] http://wordnet.princeton.edu/

Thus , the first payment should be on or after the date on which the Deed of Covenant is signed . You cannot simply draw up a covenant (or a Deposited Covenant Agreement) to cover a donation you have already made in the hope that ACET can obtain tax advantage on the sum given . However , it can be possible for the documents to be signed after you have sent a payment by cheque provided that you arrange for us to hold the cheque and not pay it into the *bank* until we have received the signed Deed of Covenant . What happens if I have difficulty in continuing to make payments ? If this unlikely situation arises , you should discuss the problem with us .

Figure 5.1. Excerpt from S3LS dataset for an instance of *bank*

"your *interest* rate." The subsequent iterations would likely acquire instances with "money paid for the use of money" sense.

Therefore, in this chapter, we evaluate the performance of the proposed graph-based algorithms against Espresso and show the effectiveness and robustness of the regularized Laplacian.

5.1. Common Experimental Settings

We evaluated the von Neumann kernel and the regularized Laplacian proposed in Chapter 4 with the task of word sense disambiguation using S3LS. We computed a pattern-instance co-occurrence matrix M from the same Equation 4.1 as Simplified Espresso. We used two types of patterns.

Unordered single words (bag-of-words) We used all single *words* (unigrams) in the provided *context* of a target instance from S3LS data sets. For example, Figure 5.1 shows the context of an instance of *bank*. Words are a set of { "Thus", ",", "the", ... }. Each word in the context represents one pattern; e.g. "Thus" is a bag-of-words pattern in the example above. Words were lowercased and pre-processed with the Porter Stemmer[5].

Local collocations A local collocation refers to the ordered sequence of tokens in the local, narrow context of the target word. We allowed a pattern to

[5]http://tartarus.org/~martin/PorterStemmer/def.txt

Table 5.1. Recall of predicted labels of *bank*

algorithm	MFS	others
Simplified Espresso	100.0	0.0
Espresso	100.0	30.2
Espresso (optimal stopping)	94.4	67.4
von Neumann kernels	92.1	65.1
regularized Laplacian	92.1	62.8

have wildcard expressions like "sale of * *interest* in * *" for the target word *interest*. We set the window size to ± 3 by a preliminary experiment.

Diffusion factor α was set to 10^{-5} for the von Neumann kernel and 10^{-2} for the regularized Laplacian, respectively.

5.2. Experiment 1: Reducing Semantic Drift

We test the von Neumann kernels and the regularized Laplacian on the same task as we used in Section 4.1.3; i.e., word sense disambiguation of word "bank." During the training phase, a pattern-instance matrix M was constructed using the training and testing data from Senseval-3 Lexical Sample (S3LS) Task. The (i, j) element of M of both kernels is set to pointwise mutual information of a pattern i and an instance j, just the same as in Espresso.

Table 5.1 illustrates how well the proposed methods reduce semantic drift, just the same as the experiment of Figure 4.3 in Section 4.1.3. We evaluate the recall on predicting the most frequent sense (MFS) and the recall on predicting other less frequent senses (others). For Espresso, two results are shown: the result on the seventh iteration, which maximizes the performance (Espresso (optimal stopping)), and the one after convergence. As in Section 4.1.3, if semantic drift occurs, recall of prediction on the most frequent sense increases while recall of prediction on other senses declines. Even Espresso was affected by semantic drift, which is again a consequence of the inherent graphical nature of Espresso-like bootstrapping algorithms. On the other hand, both proposed methods succeeded to balance the most frequent sense and other senses. Espresso at the optimal number of iterations achieved the best performance. Nevertheless, the number of iterations has to be estimated separately.

Table 5.2. Accuracy of WSD algorithms

algorithm	noun	all
most frequent sense	54.5	55.2
HyperLex [57]	64.6	—
PageRank [3]	64.5	—
Simplified Espresso	44.1	42.8
Espresso	46.9	59.1
Espresso (optimal stopping)	66.5	63.6
von Neumann kernels ($\alpha = 10^{-5}$)	**67.2**	64.9
regularized Laplacian ($\alpha = 10^{-2}$)	67.1	**65.4**

5.3. Experiment 2: WSD Benchmark Data

In order to show the effectiveness of the two kernels, we conducted experiments on the task of word sense disambiguation of S3LS data, this time not just on the word "bank" but on all target nouns in the data, following [3]. We report the results of Espresso both after convergence, and with its optimal number of iterations to show the upper bound of its performance.

Table 5.2 compares proposed methods with Espresso with various configurations. The proposed methods outperform by a large margin the most frequent sense baseline and both Simplified- and Espresso (after convergence and optimal stopping). This means that the proposed methods effectively prevent semantic drift.

Also, Espresso without early stopping shows more or less identical performance to Simplified Espresso. It is implied that the heuristics of filtering and early stopping is a crucial step not to select generic patterns in Espresso, and the result is consistent with the experiment of convergence process of Espresso in Section 4.1.3.

Espresso terminated after the seventh iteration (Espresso (optimal stopping)) is comparable to the proposed methods. However, in bootstrapping, not only the number of iterations but also a large number of parameters must be adjusted for each task and domain. This shortcoming makes it hard to adapt bootstrapping in practical cases. One of the main advantages of the proposed methods is that they have only one parameter α and are much easier to tune.

It is suggested in Sections 4.1.3 and 4.2.1 that Espresso and the von Neumann kernel with large α converge to the principal eigenvector of A. However, both Simplified- and Espresso are 10 points lower than the most frequent sense baseline. The reason seems to be because Espresso and the von Neumann kernels use pointwise mutual information as a weighting factor so that the principal eigenvector of A may not always represent the most frequent sense.[6]

We also show the results of previous graph-based methods [3], based on HyperLex [57] and PageRank [10]. These methods create an instance-pattern co-occurrence graph and extract hubs from the graph, and then apply a WSD algorithm based on maximum spanning tree. They use two graph-based algorithms, HyperLex and PageRank to extract hubs, and report that these algorithms are comparative to state-of-the-art supervised algorithms. However, these methods have seven parameters to tune in order to achieve the best performance, and hence are difficult to optimize.

5.4. Experiment 3: Sensitivity to a Different Diffusion Factor

Figure 5.2 shows the performance of the von Neumann kernels with a diffusion factor α. As expected, smaller α leads to relatedness to seed instances, and larger α asymptotically converges to the HITS authority ranking (or equivalently, Simplified Espresso).

One of the disadvantages of the von Neumann kernels over the regularized Laplacian is their sensitivity to parameter α. Figure 5.3 illustrates the performance of the regularized Laplacian with a diffusion factor α. The regularized Laplacian is stable for various values of α, while the von Neumann kernels change their behavior drastically depending on the value of α. However, α in the von Neumann kernels is upper-bounded by the reciprocal $1/\lambda$ of the principal eigenvalue of A, and the derivatives of kernel matrices with respect to α can be used to guide systematic calibration of α (see [24] for detail).

[6] A similar but more extreme case is described in [24] in which the use of a normalized weight matrix M results in an unintuitive principal eigenvector.

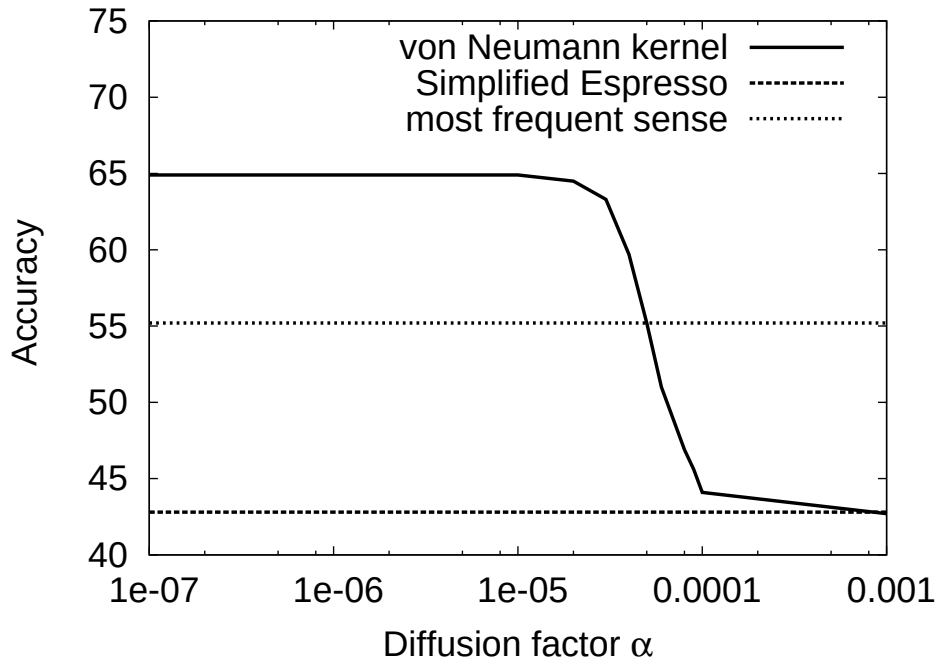

Figure 5.2. Accuracy of the von Neumann kernels with a different diffusion factor α on S3LS WSD task

5.5. Discussion

This chapter gives a graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms. We indicate that semantic drift in bootstrapping is a parallel to topic drift in HITS. We confirm that the von Neumann kernels and the regularized Laplacian reduce semantic drift in the Senseval-3 Lexical Sample task. Our proposed methods have only one parameters and are easy to calibrate.

It seems that semantic drift is likely to occur in the tasks of word sense disambiguation and semantic category acquisition since patterns are more ambiguous and abundant in these tasks than in the tasks such as relation extraction. Because bootstrapping algorithms converge after several tens of iterations, it is hard to calibrate parameters for each tasks. Thus, graph-based similarity measures are supposed to be more effective than bootstrapping algorithms in the tasks of word sense disambiguation and semantic category acquisition. In fact, Li et al. (2008) [32] report that graph-based methods constantly outperform self-training (bootstrapping) in the task of search query

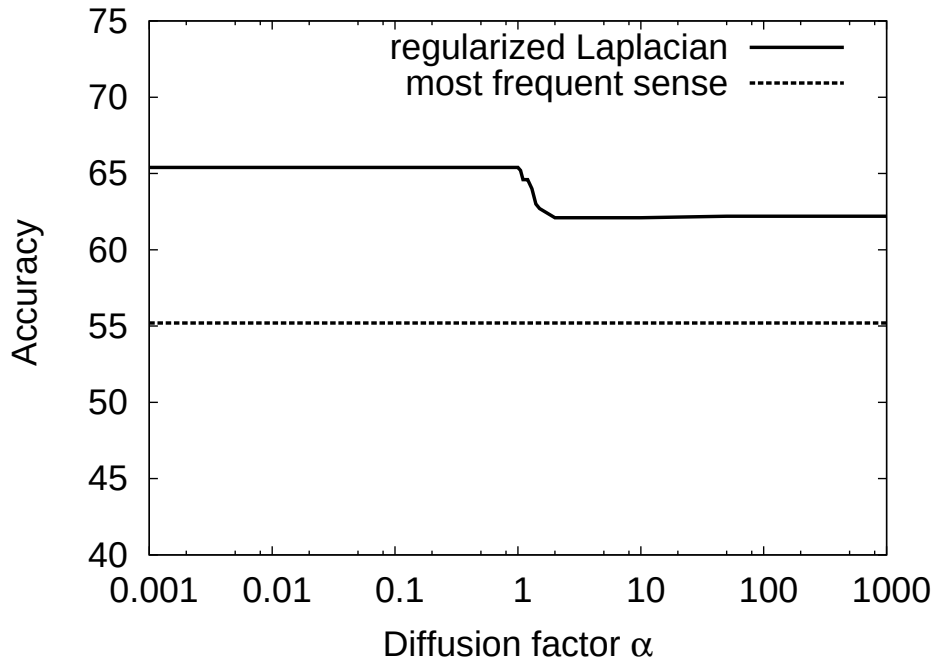

Figure 5.3. Accuracy of the regularized Laplacian with a different diffusion factor α on S3LS WSD task

classification, which is similar to the task of semantic category acquisition.

On the other hand, semantic drift may not occur in the tasks such as relation extraction where patterns are not ambiguous and only few patterns are needed for extracting instances, because a co-occurrence graph tends to be sparse. In this chapter, we demonstrate that two graph kernels to compute relatedness between nodes prevent semantic drift in the task of word sense disambiguation, where semantic drift is likely to occur. However, importance based methods might be enough depending on the structure of graphs. We will examine this hypothesis in Chapter 7.

6

Bilingual Dictionary Construction

Translation knowledge acquisition is one of the central research topics of machine translation. However, translation knowledge acquisition often depends on human annotation. Especially, annotation of technical terms requires domain knowledge. So far, reducing the cost of human annotation is one of the important problems for building machine translation systems.

To minimize the cost of hand-tagging resources, Wikipedia has been studied as a source of bilingual lexicon extraction. Wikipedia is a multilingual free online encyclopedia which is maintained by a community of volunteers. Currently, the English Wikipedia is the largest one with 2,297,611 articles, while the Japanese is the fifth place with 479,908 articles. More than 200,000 articles have both English and Japanese versions, and can easily be aligned by interlingual hyperlinks of Wikipedia. However, naïve extraction results in noisy bilingual lexicon, because Wikipedia has many ambiguous titles such as "1453 (English)" pointing to "1453 年 [*year*] (Japanese)". Also, domain adaptation of the extracted lexicon is a key issue since Wikipedia is a general-purpose encyclopedia.

Adafre and Rijke [2] proposed to use interlingual links in Wikipedia articles to obtain a bilingual lexicon. Their approach of using the interlingual links is straightforward. For each Wikipedia page in one language, they extracted interlingual hyperlinks as translations of the titles in other languages. Although they addressed the problem of ambiguous translations, they did not seem to disambiguate translation pairs since their aim was to find similar sentences instead of learning a high-quality bilingual lexicon.

Recently, Erdmann [19] showed that a link structure (combination of redirect page and link text information) can boost recall of bilingual lexicon ex-

traction from Wikipedia. However, their method does not improve accuracy of bilingual lexicon extraction, and they did not evaluate the quality of the extracted results on machine translation system. The problem of selecting appropriate word sense still exists.

In this chapter, we focus on extraction of a bilingual lexicon from Wikipedia. We followed the *one sense per domain* assumption described in Section 6.1 and extract the most likely translation pairs for each domain. We apply the *regularized Laplacian* described in Chapter 4 to the task of finding the most relevant translation pairs to the domain at hand. Graph-based methods have attracted attention in NLP tasks recently, such as word sense disambiguation [28], knowledge acquisition [55] and language modeling [50].

Our work (1) extracts a bilingual lexicon from Wikipedia and measure its quality on machine translation task, and (2) refines a bilingual lexicon based on the graph structure of Wikipedia.

The rest of this chapter is organized as follows. We propose a graph-based algorithm to disambiguate translation pairs in Section 6.1. In Section 6.2 we explain the experimental settings of our system for Patent Translation Task at NTCIR-7 [22]. We evaluate and discuss experimental results in Section 6.3.

6.1. Extraction of Domain Specific Bilingual Lexicon

Figure 6.1 depicts the overview of our extraction algorithm. The algorithm constructs a bipartite graph from Wikipedia and computes similarity between translation pairs over the graph. It requires only a small amount of translation pairs to disambiguate ambiguous translation pairs in a bilingual lexicon. It ranks a bilingual lexicon according to the similarity measure given seed translation pairs in a given domain.

First, we follow the steps described in [2] and extract a bilingual lexicon from Wikipedia. Wikipedia provides a vast number of named entities and technical terms. Some articles are associated with interlingual links. An interlingual link in Wikipedia is a link between two articles For example, [[en:Manga]] points to the English version of the article "Manga," which has an outgoing link to the Japanese version [[ja:漫画]] (*manga*). Translations of a page title (typically a noun phrase) are then given as the interlingual hyperlinks from that page. By taking the intersection of obtained list of translation pairs in both directions [1] one can obtain a large bilingual lexicon in reasonable

[1]Some page titles can not be translated back to their original ones (e.g. ambiguous words),

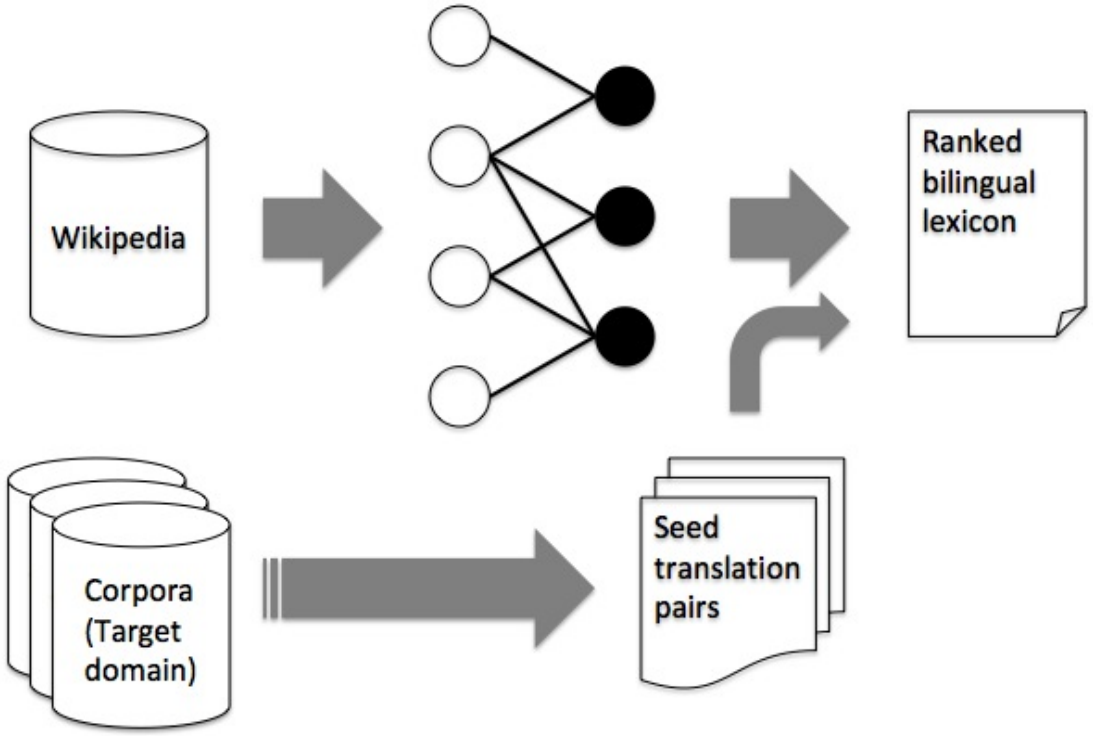

Figure 6.1. Overview of extraction of a bilingual lexicon of a target domain from Wikipedia

quality.

There are cases where a word has more than one sense. In such a case, Wikipedia provides a special page called *disambiguation* to help disambiguate word senses. One of the main problem in machine translation is to select which word sense is appropriate for a given context. We assume the *one sense per domain* hypothesis [56] by exploiting the fact that the distribution of word senses is highly skewed depending on domains. According to the hypothesis, the task of selecting the right sense is then to select the most relevant sense to the given domain.

To calculate the relatedness of a translation pair to a given domain, we manually prepare seed translation pairs from the domain and measure similarity between a translation pair and the seeds. The similarity is computed over a bipartite graph created from interlingual links and abstracts of Wikipedia[2]

and thus the two sets of translation pairs are not necessarily identical.

[2]Abstracts are automatically generated by taking the first paragraph of each page. The

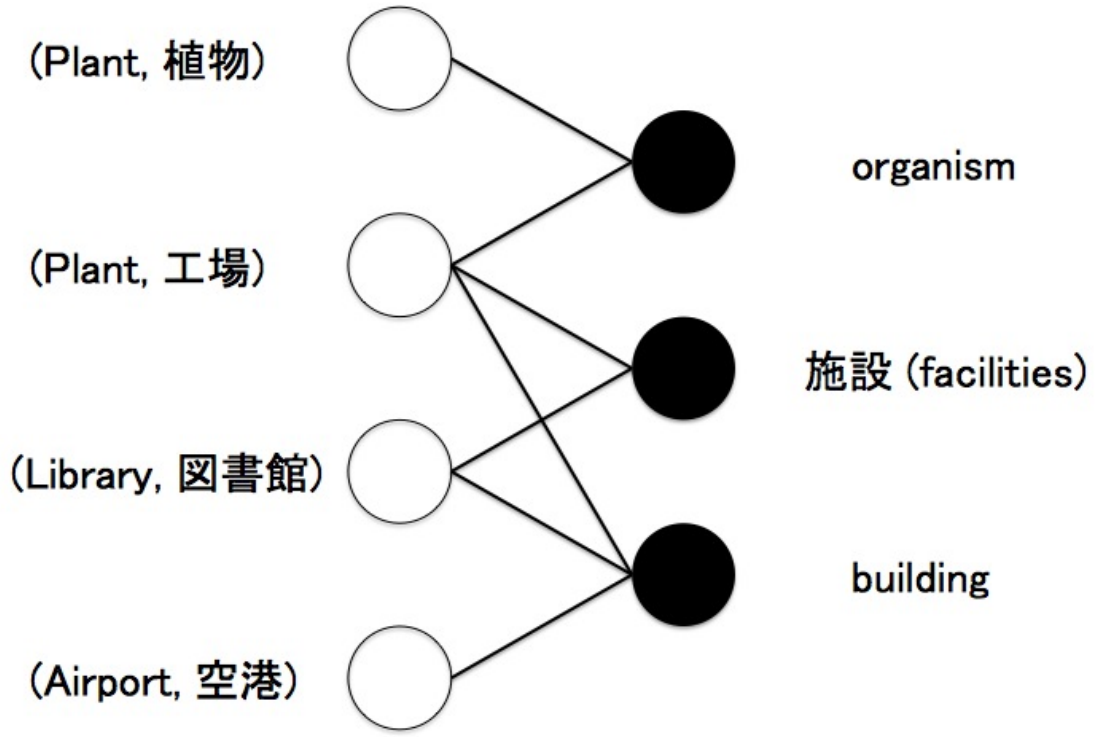

Figure 6.2. Bipartite graph from Wikipedia link structure

Figure 6.2 illustrates a bipartite graph constructed from Wikipedia. In this bipartite graph, related translation pairs tend to connect to similar set of patterns (e.g. (Library, 図書館) is more similar to (Plant, 工場) than (Plant, 植物) because they share two patterns, "施設" *facilities* (a term occurring in Japanese abstract) and "building" (a term occurring in English abstract)), and vice versa.

The steps for constructing a bipartite graph is defined as follows:

1. Add *translation pairs* (en,ja) as white nodes.

2. Add bag-of-content words (hereafter referred to as *patterns*) appearing in abstracts of both languages as black nodes. Note that a pattern may be either single English or Japanese word.

3. Add edges from translation pairs to co-occurring patterns.

abstract file (abstract.xml) is distributed as part of dump files of Wikipedia database and can be downloaded at http://download.wikimedia.org/.

The intuition behind the bipartite graph construction is that if two translation pairs share similar patterns they must be related. In the experiment below, we used pointwise mutual information defined in Section 3.2.3 as a score of an edge weight between a translation pair and a pattern.

Second, we estimate similarity between translation pairs in a bilingual lexicon by the regularized Laplacian described in Chapter 4. Hand-picked translation pairs are used for the seed vector i_0, and the pattern-instance co-occurrence matrix M is constructed as in Figure 6.1. The final instance score vector i is regarded as a vector representing the strength to the domain.

Seed translation pairs are expected to be the representative of the domain, and thus should co-occur with domain-specific patterns. To fulfill this requirement, frequent but unambiguous translation pairs should be carefully selected.

6.2. Experiment

6.2.1 Corpus and Tools

We used the first Patent Parallel Corpus (PPC-1) from NTCIR-7 Patent Translation Task for the experiment. We only used Parallel Sentence Data (PSD). The data is treated as a simple list of parallel sentences. No context and structural information which could be obtained from the Parallel Patent Data (PSD) is used.

The PSD of PPC-1 comes in four files: a training data file of about 1.8 million parallel sentences `train.txt` and three development data files of about a thousand parallel sentences `{dev,devtest,test}.txt`. We used `train.txt` for training, `dev.txt` for parameter tuning, and `test.txt` for testing during development.

We used an open source statistical machine translation system Moses[3] as a baseline system for NTCIR-7. We basically followed the instructions written at the homepage of WMT2008[4] to built the baseline system for their shared task. We build the language model by using the SRI Language Modeling Toolkit[5].

[3] http://www.statmt.org/moses/
[4] http://www.statmt.org/wmt08/
[5] http://www.speech.sri.com/projects/srilm/

6.2.2 Preprocessing and Evaluation

English sentences are tokenized and lowercased by using `tokenizer.perl` and `lowercase.perl`, the scripts provided by the WMT2008 organizers. As for Japanese sentences, its encoding is first converted from EUC-JP to UTF-8 and they are normalized under NFKC by using the Perl library. They are then word segmented by using the open source Japanese morphological analyzer MeCab[6].

In normalization form NFKC, Compatibility Decomposition and Canonical Composition is performed to unicode string. In Japanese, it roughly means double width alphabets and numbers are converted to single width, and single width Katakanas are converted to double width.

Before building the translation model, long sentences with more than 80 words are removed by using the script `clean-corpus-n.perl`. This reduces the number of training sentence pairs from 1,798,571 to 1,768,853. Both translation model and language model are made from the resulting bilingual sentences pairs.

For English outputs, detokenization is done by the script `detokenizer.perl`. Recaser is trained by using Moses from the English side of the training sentences as described in the WMT2008 baseline system. BLEU score is computed by the script `doc_bleu.rb` provided by the NTCIR-7 organizers.

6.2.3 Bilingual Lexicon Extraction

The use of a bilingual lexicon from Wikipedia described in Section 6.1 is straightforward. We add the extracted bilingual lexicon to the training corpus to learn the translation probability between translation pairs.[7]

A snapshot of Wikipedia was taken on 12 March 2008. The page titles are aligned by interlingual hyperlinks and 222,739 translation pairs are extracted in total. Non-Japanese nor English characters such as Arabic and Cyrillic are removed. Most of the formatting information which is not relevant for the current task are discarded. Eventually, 197,770 translation pairs are retained for the full Wikipedia bilingual lexicon.

We randomly split the bilingual lexicon into 8 sub lexicons (due to memory limits). 5 seeds are manually chosen for each sub lexicon (total $8 \times 5 = 40$

[6] http://mecab.sourceforge.net/

[7] One of the common ways of using a dictionary in GIZA++ is to include it as additional training data. See articles in Moses mailing list http://article.gmane.org/gname.comp.nlp.moses.user/921 for detail.

Table 6.1. Sample seed translation pairs

English	Japanese
thermal spray	溶射
epoxy	エポキシ樹脂
single crystal	単結晶
laser cooling	レーザー冷却
centrifugal compressor	遠心式圧縮機

seeds). Sample seed translation pairs are displayed in Table 6.1.

After applying the regularized Laplacian kernel, the top 10%, 50% and 75% of the ranked list for each sub lexicon are collected. The intersection of the 8 collected lists is the 10%, 50% and 75% bilingual lexicons, respectively. Table 6.2 shows the number of translation pairs for each bilingual lexicon, along with several examples (coverage of unknown words in the test corpus is also shown in parenthesis).[8]

6.2.4 Results

Performance is evaluated based on BLEU (Bilingual Evaluation Understudy) score [44]. BLEU algorithm uses n-gram overlaps between a candidate translation and a reference translation. It is a weighted precision to measure translation quality in terms of fluency and adequacy.

Table 6.3 presents BLEU scores for each translation direction for Patent Translation Task at NTCIR-7. The results of adding Wikipedia as a bilingual lexicon is shown. Wikipedia (10,50,75%) compares the effect of the graph-based refinement of the bilingual lexicon. *Formal run* stands for intrinsic evaluation for the formal run which uses multiple reference sentences, while *Single reference* uses the reference sentence distributed with the fmlrun-int dataset to compute BLEU score.

[8]The total number of words for each ranked lexicon does not necessarily proportional to the full Wikipedia since there are duplicates in the split sub lexicons.

Table 6.2. Samples of the extracted bilingual lexicon from Wikipedia

.

Wikipedia	♯ of words	samples
10%	11,970 (1.9%)	(natural selection, 自然選択説), (scrabble, スクラブル) , (phase transition, 相転移), (diamond, ダイアモンド), (videocassette recorder, ビデオテープレコーダ)
50%	75,420 (7.7%)	(movement for multiparty democracy, 複数政党制民主主義運動), (fentanyl, フェンタニル) [an opioid analgestic], (sigma sagittarii, ヌンキ) [the second brightest star system in the constellation Sagittarius]), (shintaro abe, 安倍晋太郎) [the former prime minister of Japan], (nippon television, 日本テレビ放送網)
75%	113,277 (11.5%)	(pride final conflict 2003, pride grandprix 2003 決勝戦) [a mixed martial arts event held by PRIDE Fighting Championships], (uglyness, 醜), (palma il vecchio, パルマ・イル・ヴェッキオ) [an Italian painter], (jean gilles, ジャン・ジル) [a French composer; a French soldier], (amiloride, アミロライド) [a potassium-sparing diuretic]
100%	197,770 (13.5%)	(brilliant corners, ブリリアント・コーナーズ) [an album by a jazz musician], (charly mottet シャーリー・モテ) [a French former professional cyclist], (deep purple in rock, ディープ・パープル・イン・ロック) [an album by an English rock band], (june 2003,「最近の出来事」2003年6月) [navigational entry for events happened in June 2003], (moanin', モーニン) [a jazz album]
filtered	24,969	(1, 1年) [year], (UTC+9, UTC+9) [Japanese side contains only alphanumeric characters], (Aera, AERA) [case-insensitive match] (大岡越前, 大岡越前) [garbage in English side], (image:himeji castle frontview.jpg, himeji castle frontview.jpg) [Wikipedia format navigational links], (user:eririnrinrin, eririnrinrin) [Wikipedia specific entries],

Table 6.3. BLEU score for Patent Translation Task at NTCIR-7

	Single reference		Formal run	
	JE	EJ	JE	EJ
baseline	26.39	28.25	25.34	27.19
Wikipedia (10%)	—	27.47	—	—
Wikipedia (50%)	—	27.46	—	—
Wikipedia (75%)	—	27.42	—	—
Wikipedia (100%)	26.48	27.28	25.48	28.15

6.3. Discussion

Table 6.3 demonstrates that the extracted bilingual lexicon slightly improves BLEU score (0.09 for fmlrun-int and 0.14 for official) in Japanese to English translation. However, adding the extracted bilingual lexicon constantly degrades BLEU score for fmlrun-int dataset in English to Japanese direction, while it outperforms baseline in BLEU score by 1 for the official run. It is not clear why the reported results are not consistent with the results of fmlrun-int, and thus re-examination is needed to verify the efficiency of the proposed method.

By comparing Wikipedia (75%,100%) and others, it is suggested that adding the whole bilingual lexicon extracted from Wikipedia may be too noisy to learn phrase alignments. One possibility is to extract only highly relevant terms to the domain (at the expense of coverage), and another possibility is to investigate better way to integrate a bilingual lexicon to phrase-based statistical machine translation.

In this chapter, we demonstrated that a large scale bilingual lexicon can be extracted from Wikipedia. The bilingual lexicon may be improved in its quality by a graph-based kernel. We have reported the results on NAIST-NTT system for Patent Translation Task at NTCIR-7.

Although adding a dictionary to a training corpus has the advantage of simplicity, it is not the best way to incorporate word sense disambiguation into machine translation system. Carpuat et al. (2007) [12] showed that reranking of the phrase table improves performance of statistical machine translation. It is one of the future work to integrate graph-based word sense disambiguation into statistical machine translation framework.

We have seen that we can create a graph from the interlink structure ex-

tracted from Wikipedia. However, the performance of graph-based algorithms depends on the structure of graph. In the next chapter, we investigate another resource to build a graph structure suitable for knowledge acquisition with graph kernels. We will see that by choosing appropriate source, importance-based measure is enough for the task of semantic category acquisition. It is an open question what resources are relevant for bilingual dictionary construction, especially in the context of web data mining.

7

Learning Semantic Categories

Compared to other text resources, search queries more directly reflect search users' interests [52]. Web search logs are getting a lot more attention lately as a source of information for applications such as targeted advertisement and query suggestion. However, it may not be appropriate to use queries themselves because query strings are often too heterogeneous or inspecific to characterize the interests of the user population. Although it is not clear that query logs are the best source of learning semantic categories, all the previous studies using web search logs rely on web search query logs.

Therefore, we propose to use web search clickthrough logs to learn semantic categories. The term *semantic categories* is used in prototype theory in the field of cognitive linguistics. It is not a classical category defined in terms of a necessary and sufficient condition. Rather, it is a gradable concept which is characterized as a typical instance and relatedness to the instance. [48] For instance, a typical instance for a semantic category of "bird" is a crow or a sparrow whereas a peripheral instance would be an ostrich or a penguin. In this way, the semantic category of "bird" is part of a gradable classification system. For web search, it is important to learn a semantic category that emerges everyday in search query logs.

We cast semantic category acquisition from search logs as the task of learning labeled instances from few labeled seeds. To our knowledge this is the first study that exploits search clickthrough logs for semantic category learning.

There are many techniques that have been developed to help elicit knowledge from query logs. These algorithms use contextual patterns to extract a category or a relation in order to learn a target *instance* which belongs to the category (e.g. *cat* in *animal* class) or a pair of words in specific relation (e.g.

headquarter to a *company*). In this work, we focus on extracting named entities of the same class to learn semantic categories.

Paşca and Durme [39] were the first to discover the importance of search query logs in natural language processing applications. They focused on learning attributes of named entities, and thus their objective is different from ours. Another line of new research is to combine various resources such as web documents with search query logs [40, 55]. We differ from this work in that we use search clickthrough logs rather than search query logs.

Xu et al. (2009) [60] exploit search clickthrough logs to learn semantic categories. However, they model clickthrough data with Latent Dirichlet Allocation (LDA) [7] and use a label of semantic categories of seed instances as training data to estimate semantic categories by probabilistic model.

Also, Li et al. (2008) [32] learn query intent from search clickthrough logs. They build an instance similarity graph from search clickthrough logs and use label propagation. However, they do not use graph Laplacian and their task is different from ours.

As we have seen in Chapter 3, Tchai is dedicated to the task of semantic category acquisition from search query logs. It achieves state-of-the-art performance for this task, but it only uses web search query logs. Also, the Tchai algorithm does not scale to large corpora.

Resource. Because web search queries are heterogeneous and highly ambiguous, they do not necessarily reflect search users' intent. Thus, it might be inappropriate to use web search query logs as patterns. However, most of the previous work uses web search query logs.

Scalability. Espresso-style algorithms including Tchai described in Chapter 3 need more than 8 parameters such as the number of seed instances, stopping criterion, the number of instances and patterns to select on each iteration, and so on, to calibrate. Moreover, because bootstrapping algorithms are sensitive to these parameters [36], it is necessary to hand-tune optimal parameters to apply to real tasks.

Therefore, we propose two solutions for each problem in this chapter.

Exploiting *search clickthrough logs*. First, we tackle the problem of resource by using search clickthrough logs instead of search query logs. Joachims [25] developed a method that utilizes clickthrough logs for training ranking of

search engines. A *search clickthrough* is a link which search users click when they see the result of their search. The intentions of two distinct search queries are likely to be similar, if not identical, when they have the same clickthrough. Search clickthrough logs are thus potentially useful for learning semantic categories. Clickthrough logs have the additional advantage that they are available in abundance and can be stored at very low cost.[1] Our proposed method employs search clickthrough logs to improve semantic category acquisition in both precision and recall.

Semi-supervised learning with *Laplacian label propagation*. Second, we solve the problem of scalability by applying Laplacian label propagation described in Section 4.2.3 to the task of semantic category acquisition. Laplacian label propagation reduces 8 parameters of Tchai to only 1 parameter (diffusion factor) while achieving comparative accuracy. Graph-based methods such as label propagation have additional advantage that they are easy to apply distributed computing. Thus, graph-based methods are desirable to the task of mining semantic knowledge from search clickthrough logs since clickthrough logs tend to be very sparse.

As far as we know, we are the first to exploit search clickthrough logs by using the regularized Laplacian for the task of semantic category acquisition.

7.1. Quetchup Algorithm

In this section, we describe an algorithm for learning semantic categories from search logs using label propagation. We name the algorithm *Quetchup*.

Label propagation methods process an instance similarity matrix A. It is not unusual to deal with more than 10 millions of magnitude of instances when we extract semantic knowledge from web resources. However, a typical configuration of workstations which have few gigabytes can only perform the eigenvalue decomposition of a matrix of size not more than several tens of thousands, and it is not possible to handle large scale data unless some dimension reduction technique is used. Therefore, it is necessary to manage large scale data efficiently.

[1] As for data availability, MSN Search query logs (RFP 2006 dataset) were provided to WSCD09: Workshop on Web Search Click Data 2009 participants. http://research.microsoft.com/en-US/um/people/nickcr/WSCD09/

7.1.1 Reducing Data Size

First, similarity matrix A is a dense matrix which requires the size of $O(n^2)$ storage, and it is unrealistic to hold it when the similarity matrix is large. For instance, suppose we hold similarity information in 4 bits. Even so, it can only afford a matrix of size no more than hundreds of thousands with 16 gigabytes of memory, which is not realistic setting nowadays. Therefore, we divide similarity matrix A into two components, $A = W^T W$, and store only instance-pattern matrix W to keep the size of data small. While A is a dense matrix of $n \times n$ size, instance-pattern matrix W is a sparse matrix which requires only the size of $O(np)$ storage where p is the average number of patterns per instance. Typically p is $n \gg p$ and less than hundreds, so the computation over the graph is manageable. [2]

7.1.2 Approximating Label Propagation

As for computation, Laplacian label propagation involves singular value decomposition and computes

$$F^* = \sum_{t=0}^{\infty} (\alpha(-\mathcal{L}))^t F(0) = (I + \alpha\mathcal{L})^{-1} F(0)$$

but the matrix inversion leads to $O(n^3)$ complexity, which is far from realistic in real-world configuration. Nonetheless, it can be approximated by fixing the number of steps for label propagation. The number of step t of label propagation corresponds to the length of paths from the current node to propagated nodes. All the paths are taken into consideration as t increases, while only neighbouring nodes are taken into account as t approaches 0. In particular, $t = 0$ is a special case that using only seed labels for classification. Also, $t = 1$ means propagating seed labels to instance nodes which co-occur with the pattern nodes that occur with seed instances. In other words, it takes second order co-occurrence into account.

There are two advantages in this approximation: (1) as we have just described in this section, the computation of label propagation is reduced to $O(npt)$ complexity; and (2) it allows parallel and distributed computation of label propagation using MapReduce [17]. A set of map and reduce operations amounts to a single step of label propagation. The computation of $\alpha(-\mathcal{L})F(t)$

[2] One can perform dimension reduction such as random projection to further compress the size of data.

Category	Seed
Travel	jal (Japan Airlines), ana (All Nippon Airways), jr (Japan Railways), じゃらん (jalan: online travel guide site), his (H.I.S.Co.,Ltd.: travel agency)
Finance	みずほ銀行 (Mizuho Bank), 三井住友銀行 (Sumitomo Mitsui Banking Corporation), jcb, 新生銀行 (Shinsei Bank), 野村證券 (Nomura Securities)

Table 7.1. Seed terms for each category

to obtain $F(t+1)$ requires $O(np)$. MapReduce allows this computation run in parallel and the iteration process can be performed efficiently by storing $\mathcal{L}$ in a local disk to exploit locality of disks.

One of the difficulties in applying the regularized Laplacian to the real world tasks is that it is difficult to determine the amount of time necessary for finding solutions. In practice, it hardly matters whether solutions are approximate (with guaranteed precision) or not. It is often preferable to have a faster algorithm that returns solutions within a certain range, and MapReduce meets this end.

7.2. Experiments with Web Search Logs

We will describe experimental results comparing a previous method Tchai to the proposed method Quetchup with clickthrough logs ($Quetchup_{click}$) and with query logs ($Quetchup_{query}$).

7.2.1 Experimental Settings

Search logs: We used Japanese search logs collected in August 2008 from Yahoo! JAPAN Web Search. We thresholded both search query and clickthrough logs and retained the top 1 million distinct queries. Search logs are accompanied by their frequencies within the logs.

Construction of an instance-pattern matrix: We used clicked links as clickthrough patterns. Links clicked less than 200 times were removed. After that,

links which had only one co-occurring query were pruned. [3] On the other hand, we used two term queries as contextual patterns. For instance, if one has the term "singapore" and the query "singapore airlines," the contextual pattern "♯ airlines" will be created. Query patterns appearing less than 100 times were discarded.

The (i, j)-th element of a row-normalized instance-pattern matrix W is given by
$W_{ij} = \frac{|x_i, p_j|}{\sum_k |x_i, p_k|}$.

Target categories: We used two categories, Travel and Finance, to compare Quetchup with clickthrough and query logs against Tchai described in Chapter 3.

When a query was a variant of a term or contains spelling mistakes, we estimated original form and manually assigned a semantic category. We allowed a query to have more than two categories. When a query had more than two terms, we assigned a semantic category to the whole query taking each term into account.[4]

System: We used the same seeds presented in Table 7.1 for both Tchai and Quetchup. We used the same parameter for Tchai described in Chapter 3 and collected 100 instances by iterating 10 times and extracting 10 instances per iteration. The number of iteration of Quetchup is set to 10. The parameter α is set to 0.0001.

Evaluation: It is difficult in general to define recall for the task of semantic category acquisition since the true set of instances is not known. Thus, we evaluated all systems using *precision at k* and *relative recall* [43].[5] Relative recall is the coverage of a system given another system as baseline.

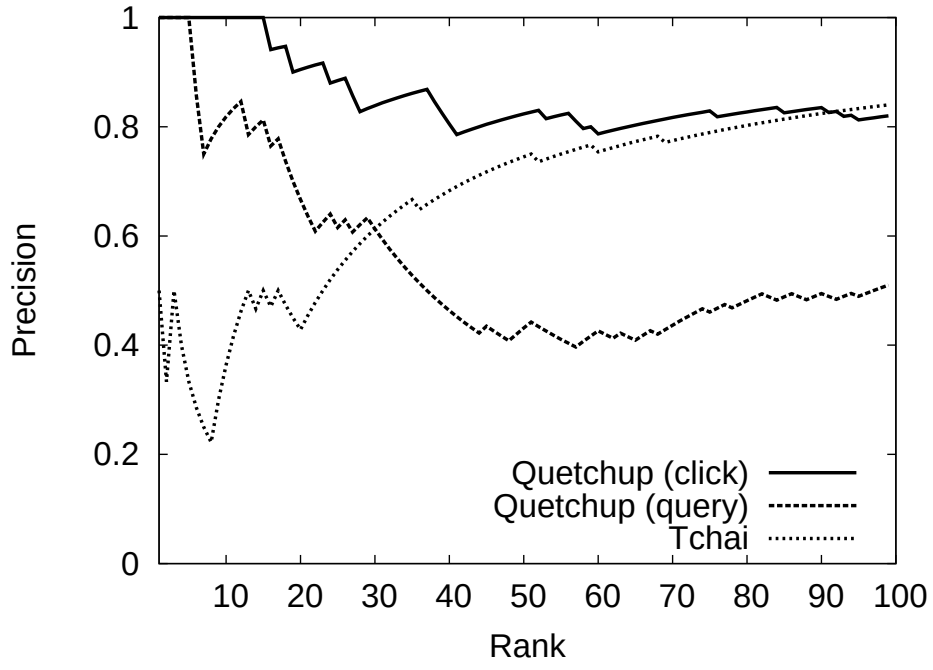

Figure 7.1. Precision of Travel domain

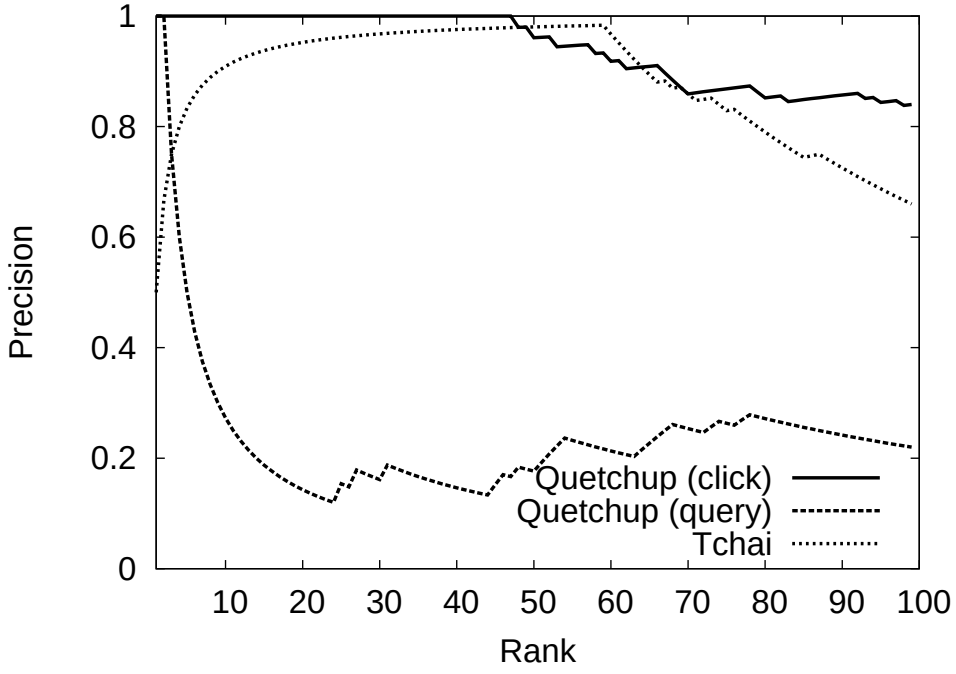

Figure 7.2. Precision of Finance domain

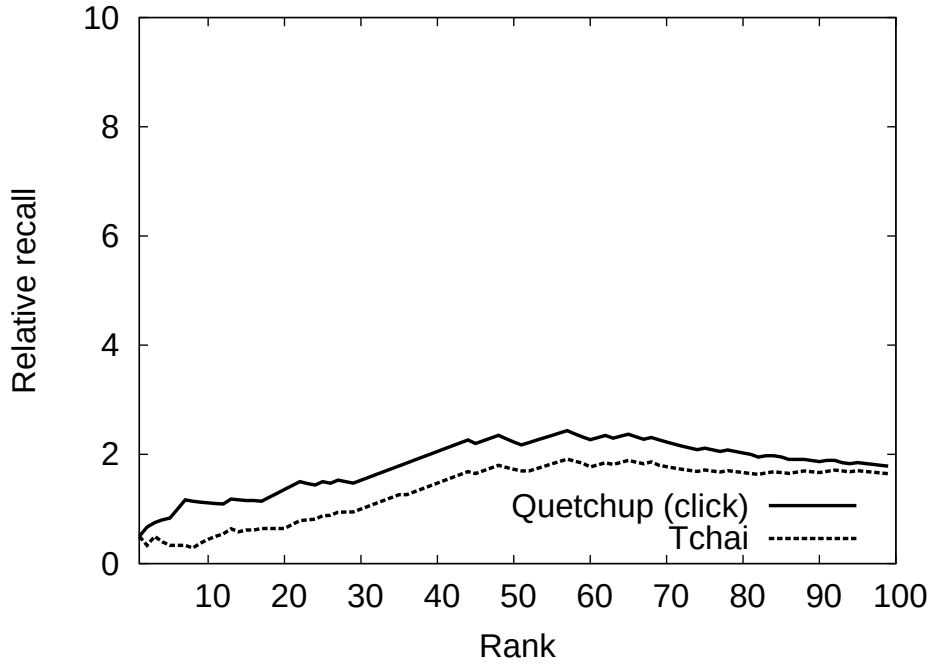

Figure 7.3. Relative recall of Travel domain

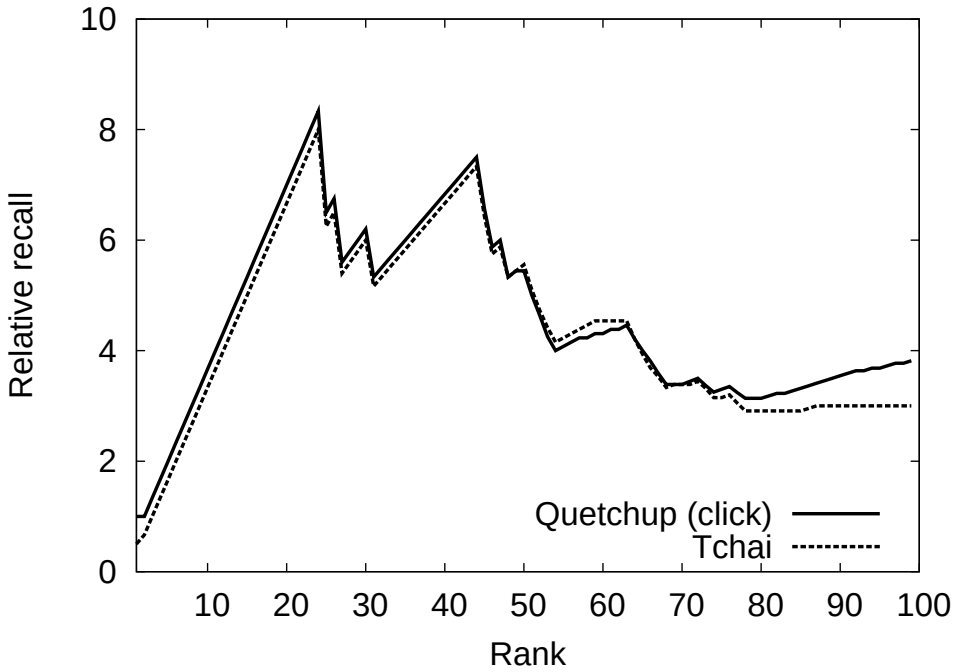

Figure 7.4. Relative recall of Finance domain

7.2.2 Results

Figures 7.1 to 7.4 plot precision and relative recall for three systems to show effectiveness of search clickthrough logs in improvement of precision and relative recall. Relative recall of Quetchup$_{click}$ and Tchai were calculated against Quetchup$_{query}$.

Quetchup$_{click}$ gave the best precision among three systems, and did not degenerate going down through the list. In addition, it was demonstrated that Quetchup$_{click}$ gives high recall. This result shows that search clickthrough logs effectively improve both precision and recall for the task of semantic category acquisition.

On the other hand, Quetchup$_{query}$ degraded in precision as its rank increased. Manual check of the extracted queries revealed that the most prominent queries were Pornographic queries, followed by Food, Job and Housing, which frequently appear in web search logs. Other co-occurrence metrics such as pointwise mutual information would be explored in the future to suppress the effect of frequent queries.

In addition, Quetchup$_{click}$ constantly outperformed Tchai in both the Travel and Finance domains in precision and outperformed Quetchup$_{query}$ in relative recall. The differences between the two domains of query-based systems seem to lie in the size of correct instances. The Finance domain is a closed set which has only a few effective query patterns, whereas Travel domain is an open set which has many query patterns that match correct instances. Quetchup$_{click}$ has an additional advantage that it is stable across over the ranked list, because the variance of the number of clicked links is small thanks to the nature of the ranking algorithm of search engines.

7.2.3 Mixing Clickthrough and Query Logs

In the experiment of Section 7.2.2, we make a matrix using both clickthrough and query patterns. Instance-pattern matrices of clickthrough and query patterns are normalized and mixed by a parameter $\beta \in [0, 1]$.

[3]Pruning facilitates the computation time and reduces the size of instance-pattern matrix drastically.

[4]Since web search query logs contain many spelling mistakes, we experimented in a realistic configuration.

[5]Typically, precision at k is the most important measure since the top k highest scored terms are evaluated by hand.

$$W_{mix} = \beta W_{click} + (1 - \beta) W_{query}$$

where W_{mix} is also a row-normalized instance-pattern matrix. Similarity matrix A_{mix} using clickthrough and query patterns is given as follows:

$$A_{mix} = W_{mix}^T W_{mix}.$$

The ratio of clickthrough and query logs, β, was changed from 0 to 1 by 0.2. $\beta = 0.2$ means that to the ratio of clickthrough:query is 2:8. Also, relative recall of *Quetchup (click)* and *Tchai* was calculated against *Quetchup (query)*.

Figures 7.5 and 7.6 show that when the ratio of clickthrough logs are high, precision and relative recall increase. The system using only clickthrough logs give the best precision. The line "click:query=100:0" in Figure 7.6 indicates that relative recall of Quetchup (click) is 1.5-2 times higher than the system using query logs, and confirms that clickthrough logs not only gives high precision but also high recall.

7.2.4 Extracted Instances and Patterns

Table 7.2 displays the instances and patterns that had the top 10 highest scores to show characteristics for search query and clickthrough logs.

Thanks to clickthrough logs, Quetchup$_{click}$ succeeded to learn language variations and spelling mistakes as well as semantic categories. The reason behind this is that search engines are so robust to language variations and spelling mistakes that they can assign high ranks to relevant pages to a query.

On the other hand, label propagation with search query learned various queries other than synonyms, at the cost of irrelevant queries such as "adbox" and "アダコミ," and both precision and relative recall fell down.

Table 7.3 classifies random sampled 100 instances from the 10,000 highest scored queries using the top 10 million clickthrough logs.

Over the half of the queries, the most frequent queries are the queries related to transportation. This seems because three out of five seed queries are related to transportation, and thus tend to extract queries related to this type.

The next frequent queries are related to accommodation and travel information. Label propagation succeeds to learn them without providing seed queries for these types.

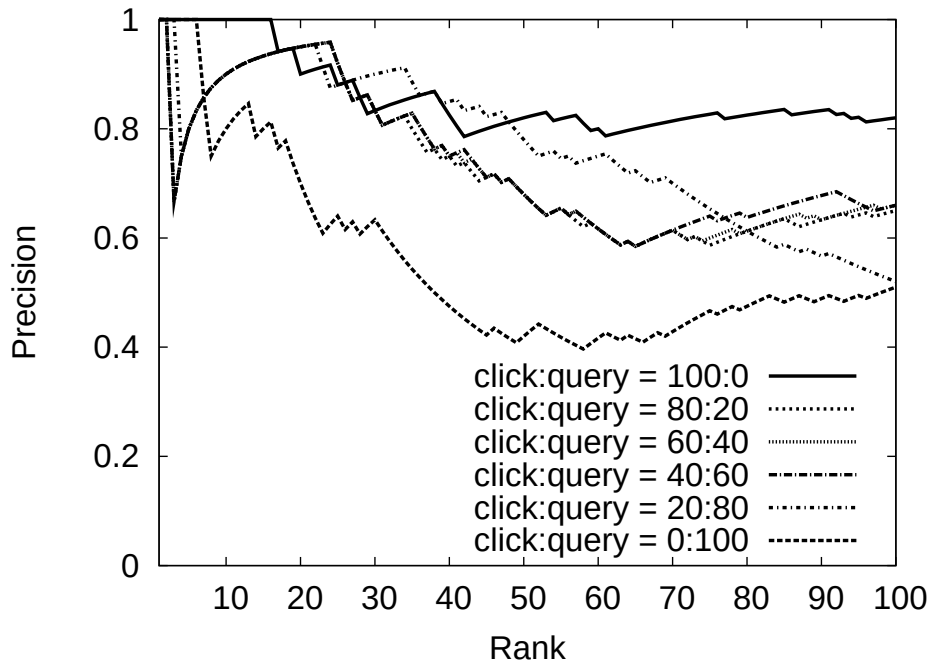

Figure 7.5. Precision of Quetchup with various Click:Query ratio

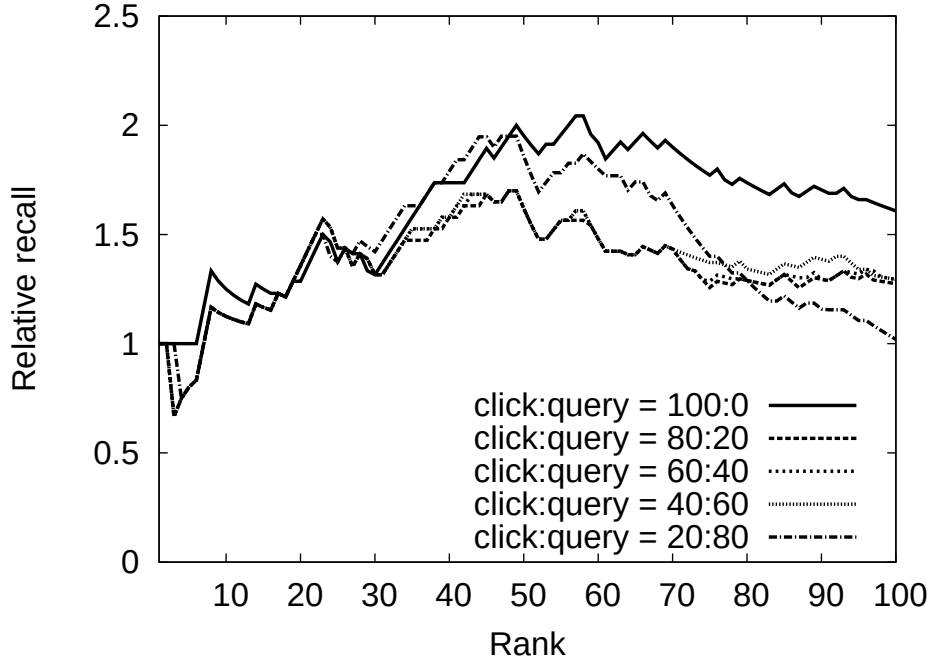

Figure 7.6. Recall of Quetchup with various Click:Query ratio

Table 7.2. Extracted Instances and Patterns with Top 10 Highest Scores

system	instance	pattern (leading http:// was omitted)
Quetchup (click)	じゃらん 宿泊 (jalan accommodation), じゃラン (jalan), ジャラン (jalan), jarann, jaran, じゃらん net (jalan.net), jalan, じゅらん (julan), ana 予約 (ana reservation), ana.co.jp	www.jalan.net/, www.ana.co.jp/, www.his-j.com/ www.jreast.co.jp/, www.jtb.co.jp/, www.jtb.co.jp/ace/, www.westjr.co.jp/, www.jtb.co.jp/kaigai/, nippon.his.co.jp/, www.jr.cyberstation.ne.jp/
Quetchup (query)	中部発 (Traveling from Midland), his 関西 (his Kansai), 伊平屋島 (Iheya Island), ホテルコンチネンタル横浜 (Hotel Continental Yokohama), げんじいの森 (Genjii-no-mori; spa), フジサファリパーク (Fuji Safari Park), ad-box, アダコミ (adacomi; offensive), スカイチーム (SkyTeam), ノースウェスト (Northwest)	♯ 時刻表 (timetable), ♯ 国内旅行 (domestic tour), ♯ 宿泊 (accommodation), ♯ 北海道 (Hokkaido), ♯ 関西 (Kansai), ♯ 九州 (Kyushu), ♯ マイレージ (mileage), ♯ 名古屋 (Nagoya), ♯ 沖縄 (Okinawa), ♯ 温泉 (spa)
Tchai (1st iter.)	jtb, 新幹線, 航空券, 全日空, 飛行機, 格安, 国内線, 旅行, 高速バス, jr 東日本	♯ キャンセル料, ♯ キャンセル, ♯ 早割り, ♯ 北海道, ♯ キャビンアテンダント, ♯ グランドスタッフ, ♯ 陸マイラー, ♯ スカイメート, ♯ 機内販売, ♯ 介護割引
Tchai (10th iter.)	静鉄バス (Seitetsu Bus), 相鉄バス (Sotetsu Bus), 函館バス (Hakodate Bus), 大阪地下鉄 (Osaka subway), 琴電 (Kotoden railways), 地下鉄御堂筋線 (Subway Midosuji Line), 芸陽バス (Geiyo Bus), 新京成バス (Shin-keisei Bus), jr 阪和線 (Japan Railways Hanna Line), 常磐線 (Joban Line)	♯ 時刻表, ♯ 路線図 (route map), ♯ 運賃 (fare), ♯ 料金 (fare), ♯ 定期 (season ticket), ♯ 運行状況 (service situation), ♯ 路線 (route), ♯ 定期代 (season ticket fare), ♯ 定期券 (season ticket), ♯ 時刻

Table 7.3. Random samples from extracted instances

type	♯	instance
Transportation	54	広島 新幹線 (Hiroshima Super Express), 東海道線 (Tokaido Line), jr 飯田線 (JR Iida Line), jr 博多 (JR Hakata), 京都 新幹線 (Kyoto Super Express)
Accommodation	10	ホテルビーナス (Hotel Venus), リーガロイヤルホテル大阪 (Rihga Royal Hotel Osaka), www.route-inn.co.jp, ホテル京阪ユニバーサル・シティ (Hotel Keihan Universal City), 札幌全日空ホテル (ANA Hotel Sapporo)
Travel information	10	外務省 安全 (Foreign Ministry safety), チケットショップ 大阪 (ticket shop Osaka), 観光 関西 (Sight seeing Kansai), 高山観光協会 (Takayama Tourism Association), グーグル ナビ (Google navi)
Travel agent	6	jr おでかけネット (Odekake net), 近畿ツー (Kinki Tourist), タビックス 静岡 (Tabix Shizuoka), フレックスインターナショナル (Flex International), オリオンツアー (Orion Tour)
Travel misc.	2	プロテカ (Proteca; bag for travel), jal 紀行倶楽部 (JAL Travel Club)
Others	20	格安航空チケット 海外 (discount flight ticket overseas), 新幹線予約状況 (Super express reservation situation), 新幹線 時刻表 (Super express timetable), 温泉宿 (spa accommodation), 新幹線 停車駅 (Super express stops), 虎 (tiger), youtubu 海外ドラマ (overseas drama), 法務部採用 (legal department recruitment), おくりびと (Okuribito; film), 社会人野球 (amateur baseball)

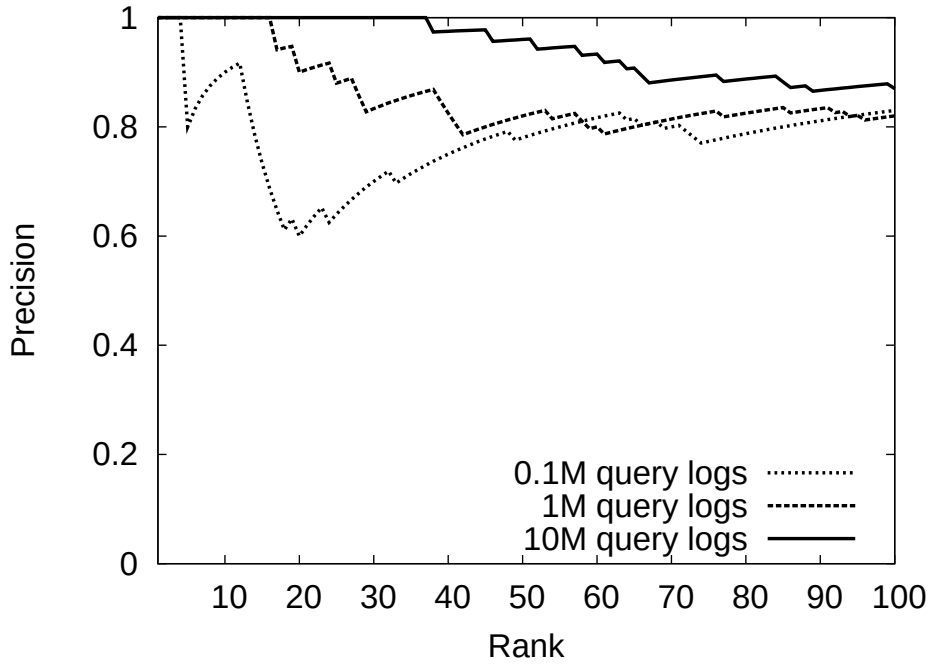

Figure 7.7. Effect of corpus size to precision

Also, 20% of the system output did not contain any named entities, but 1/4 of them were navigational queries[6] related to travel. We can conclude that only about 20% of the learned queries are out of domain even when we extract 10,000 queries, and the proposed system learns not only synonyms and spelling variations but also varieties of subtypes of semantic categories in high precision.

7.2.5 Comparison between Data Size

Figure 7.7 plots precision of the Quetchup algorithm varying the size of the corpus to see the learning curve. We used the top 0.1, 1 and 10 millions frequent search clickthrough logs.

Compared to small amount of data, large amount of data improved precision. It seems that search clickthrough logs are sparser than search query logs, and the more available data the denser a clickthrough graph becomes.

[6] A navigational query is the one that has a unique satisfactory page [11]. e.g. "YouTube" to http://www.youtube.com/.

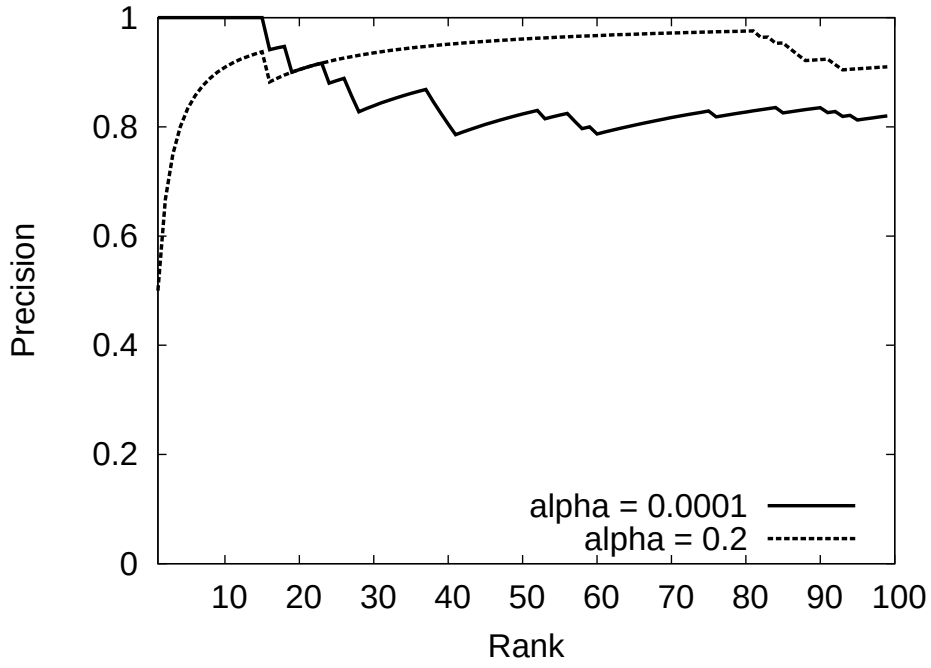

Figure 7.8. Effect of α to precision

When an instance-pattern graph is dense there are more nodes which can be reached from seed instances.

7.2.6 Performance with Varying Diffusion Factor

Figure 7.8 shows precision of the Quetchup algorithm with varying diffusion factor α, which controls how important labeled instances are. We only present the result of $\alpha = 0.2$ because the results remain almost the same with any other values larger than 0.2. It is expected that if α is small label propagation tends to a relatedness measure and semantic drift will not occur, whereas if α is large it over-emphasize graph structure and semantic drift will occur.

However, from the experiment, Quetchup with large α achieves higher precision than that with small α. This counter-intuitive result can be explained in terms of the graph structure of clickthrough logs. The reason seems that the bipartite graph using clickthrough logs is sparser than the graph using query logs, and the latter may contain erroneous edges due to insufficient information provided to search users. Densely connected components of graphs using clickthrough logs thus captures importance in Travel domain since click-

through logs reflect actual search users' behaviour.

In fact, queries used as seed instances are ranked jr (43), じゃらん（378）, ana (755), jal (904), his (1362) when $\alpha = 0.8$. As you can see, seed instances are not necessarily ranked high. Therefore, semantic drift takes place if diffusion factor is large; however, it may not degenerate precision. This means that the seed instances used for this dataset was not the best seed instances for the target semantic category.

7.3. Discussion

We have proposed a method called Quetchup to learn semantic categories from search clickthrough logs using Laplacian label propagation. The proposed method greatly outperforms previous method, taking the advantage of search clickthrough logs.

The main issue in graph-theoretic minimally-supervised learning lies in the way how to create co-occurrence (or similarity) graph. It is not trivial to obtain a good graph structure for knowledge acquisition. Not only "relatedness" metric but also "importance" metric can be exploited in natural language processing tasks if one finds an effective graph for a certain application.

8 Conclusion

This work gives a graph-based analysis of semantic drift in Espresso-like bootstrapping algorithms. We indicate that semantic drift in bootstrapping is a parallel to topic drift in HITS. We confirm that the von Neumann kernels and the regularized Laplacian reduce semantic drift in the tasks of word sense disambiguation, bilingual dictionary construction and semantic category acquisition. Our proposed methods have only one parameters and are easy to calibrate.

Mutual exclusion bootstrapping [16] uses manually specified instances causing semantic drift (stop class) to achieve bootstrapping in high precision. Because we know that the cause of semantic drift has the same root as topic drift in link analysis, we would automatically extract stop class from a patter-instance co-occurrence matrix without any supervision. Along with a recently proposed method to select seed instances [59], we can further reduce the cost of human annotation in bootstrapping algorithms.

As we have seen in Section 5.4, the deficiency of the von Neumann kernel to the regularized Laplacian is the sensitivity to diffusion factor. Recently, Kunegis et al. (2009) [30] propose an automatic way to tune parameters for graph kernels including the von Neumann kernel and the regularized Laplacian. Thanks to the graph-theoretic foundation of these kernels, we can apply the automatic parameter tuning method to our tasks. This could not be done with heuristic-based bootstrapping algorithms.

Beside the regularized Laplacian, many other kernels based on the eigenvalue regularization of the Laplacian matrix have been proposed in machine learning community [29, 34, 49]. One such kernel is the commute-time kernel [49] defined as the pseudo-inverse of Laplacian. Despite having no parame-

ters at all, it has been reported to perform well in many collaborative filtering tasks [21]. We plan to test these kernels in our task as well.

There still remains several topics to explore. We will leave the following questions open to future work.

First, we have not investigated the effect of seed instances to the performance of bootstrapping. It is known that seed instances affect the quality of bootstrapping [59] and we are not completely free from selecting "good" seeds, unless some conditions are met [18]. It is not yet clear whether prototype-based seed selection is appropriate for the task of semantic category acquisition. One alternative is to use basic level category [48] for a seed selection criterion. It is an open question what properties effective seeds share, and how to find those seeds.

Second, even though we tackled several task of bootstrapping algorithms, we have not formulated the problem of bootstrapping algorithms in terms of learning from patterns as seeds. Just like Hearst (1992) [23] starts from several seed patterns to learn hyponym relations, we could define the problem of bootstrapping algorithms to learn from seed patterns as well.

Third, it is not well studied how we can create effective graphs for NLP tasks. Although pointwise mutual information based co-occurrence metrics are used for Espresso-style algorithms, it may not be an optimal co-occurrence metric. Recently, Joachims (2002) [25] proposed a method for learning to rank using clickthrough data. Bai et al. (2009) [4] adapted a similar method to learn semantic similarity in a supervised manner. It might be worthwhile to perform supervised metric learning to generate an appropriate graph to the task at hand.

Finally, we could further investigate other semi-supervised learning techniques such as co-training [8]. As we have described in this thesis, self-training can be thought of a graph-based algorithm. It is also interesting to analyze how co-training is related to the proposed algorithm.

This thesis provides graph-theoretic approaches to natural language processing, dedicated to computational semantics. We are in the middle of transition from fully-supervised shallow natural language processing to semi-supervised deep natural language learning, with the help of large-scale web as a corpus and abundant implicit feedbacks from users as cheap supervision. We are to bridge the gap between natural language processing and natural language learning towards the goal of understanding the source of meaning.

References

[1] Steven Abney. Understanding the Yarowsky Algorithm. *Computational Linguistics*, 30(3):365–395, 2004.

[2] Sisay Fissaha Adafre and Maarten de Rijke. Finding Similar Sentences across Multiple Languages in Wikipedia. In *Proceedings of EACL 2006 Workshop: Wikis and blogs and other dynamic text source*, 2006.

[3] Eneko Agirre, David Martínez, Oier López de Lacalle, and Aitor Soroa. Two graph-based algorithms for state-of-the-art WSD. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 585–593, 2006.

[4] Bing Bai, Jason Weston, David Grangier, Ronan Collobert, Kunihiko Sadamasa, Yanjun Qi, Corinna Cortes, and Mehryar Mohri. Polynomial Semantic Indexing. In *Advances in Neural Information Processing Systems 22*, pages 64–72, 2009.

[5] Michele Banko and Eric Brill. Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics*, pages 26–33, 2001.

[6] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st ACM SIGIR Conference*, pages 104–111, 1998.

[7] David Blei, Andrew Ng, and Michael Jordan. Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[8] Avrim Blum and Tom Mitchell. Combining Labeled and Unlabeled Data with Co-Training. In *Proceedings of the Workshop on Computational Learning Theory (COLT)*, pages 92–100, 1998.

[9] Sergey Brin. Extracting Patterns and Relations from the World Wide Web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT '98*, pages 172–183, 1998.

[10] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[11] Andrei Broder. A Taxonomy of Web Search. *ACM SIGIR Forum*, 36(2):3–10, 2002.

[12] Marine Carpuat and Dekai Wu. Improving Statistical Machine Translation using Word Sense Disambiguation. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL 2007)*, 2007.

[13] Pavel Yu Chebotarev and Elena V. Shamis. On proximity measures for graph vertices. *Automation and Remote Control*, 59(10):1443–1459, 1998.

[14] Michael Collins and Yoram Singer. Unsupervised Models for Named Entity Classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100–110, 1999.

[15] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13:21–27, 1976.

[16] James R. Curran, Tara Murphy, and Bernhard Scholz. Minimising semantic drift with Mutual Exclusion Bootstrapping. In *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, pages 172–180, 2007.

[17] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *Proceedings of the 6th Conference on Symposium on Operating Systems Design and Implementation*, pages 1–13, 2004.

[18] Jason Eisner and Damianos Karakos. Bootstrapping without the Boot. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP)*, pages 395–402, 2002.

[19] Maike Erdmann. Extraction of Bilingual Terminology from the Link Structure of Wikipedia, 2008. Master's thesis, Osaka University, Osaka, Japan.

[20] Oren Etzioni, Michael Cafarella, Dong Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised Named-Entity Extraction from the Web: An Experimental Study. *Artificial Intelligence*, 165(1):91–134, 2005.

[21] François Fouss, Luh Yen, Pierr Dupont, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 19(3):355–369, 2007.

[22] Atsushi Fujii, Masao Utiyama, Mikio Yamamoto, and Takehito Utsuro. Overview of the Patent Translation Task at the NTCIR-7 Workshop. In *Proceedings of the 7th NTCIR Workshop Meeting on Evaluation of Information Access Technologies: Information Retrieval, Question Answering and Cross-lingual Information Access*, 2008.

[23] Marti Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. In *Proceedings of the Fourteenth International Conference on Computational Linguistics*, pages 539–545, 1992.

[24] Takahiko Ito, Masashi Shimbo, Taku Kudo, and Yuji Matsumoto. Application of Kernels to Link Analysis. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 586–592, 2005.

[25] Thorsten Joachims. Optimizing Search Engines Using Clickthrough Data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002.

[26] Jaz Kandola, John Shawe-Taylor, and Nello Cristianini. Learning Semantic Similarity. In *Advances in Neural Information Processing Systems 15*, pages 657–664, 2002.

[27] Jon Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632, 1999.

[28] Mamoru Komachi, Taku Kudo, Masashi Shimbo, and Yuji Matsumoto. Graph-based Analysis of Semantic Drift in Espresso-like Bootstrapping Algorithms. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 1010–1019, 2008.

[29] Risi Imre Kondor and John Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the 19th International Conference on Machine Learning (ICML-2002)*, pages 315–322, 2002.

[30] Jérôme Kunegis and Andreas Lommatzsch. Learning Spectral Graph Transformations for Link Prediction. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 561–568, 2009.

[31] Hang Li and Cong Li. Word Translation Disambiguation Using Bilingual Bootstrapping. *Computational Linguistics*, 30(1):1–22, 2004.

[32] Xiao Li, Ye-Yi Wang, and Alex Acero. Learning Query Intent from Regularized Click Graphs. In *Proceedings of SIGIR'08: the 31st Annual ACM SIGIR conference on Research and Development in Information Retrieval*, pages 339–346, 2008.

[33] Rada Mihalcea. Co-training and Self-training for Word Sense Disambiguation. In *HLT-NAACL Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 33–40, 2004.

[34] Boaz Nadler, Stephane Lafon, Ronald Coifman, and Ioannis Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck operators. *Advances in Neural Information Processing Systems 18*, pages 955–962, 2006.

[35] Hwee Tou Ng. Exempler-Based Word Sense Disambiguation: Some Recent Improvements. In *Proceedings of the Second Conference on Empirical Methods in Natural Language Processing*, pages 208–213, 1997.

[36] Vincent Ng and Claire Cardie. Weakly Supervised Natural Language Learning Without Redundant Views. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 94–101, 2003.

[37] Zheng-Yu Niu, Dong-Hong Ji, and Chew Lim Tan. Word Sense Disambiguation Using Label Propagation Based Semi-Supervised Learning. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 395–402, 2005.

[38] Marius Paşca. Organizing and Searching the World Wide Web of Fact — Step Two: Harnessing the Wisdom of the Crowds. In *Proceedings of the 16th International World Wide Web Conference (WWW-07)*, pages 101–110, 2007.

[39] Marius Paşca and Benjamin Van Durme. What You Seek is What You Get: Extraction of Class Attributes from Query Logs. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2832–2837, 2007.

[40] Marius Paşca and Benjamin Van Durme. Weakly-Supervised Acquisition of Open-Domain Classes and Class Attributes from Web Documents and Query Logs. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-2008)*, pages 19–27, 2008.

[41] Marius Paşca, Dekang Lin, Jeffrey Bigham, Andrei Lifchits, and Alpa Jain. Organizing and Searching the World Wide Web of Facts — Step One: the One-Million Fact Extraction Challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence (AAAI-06)*, pages 1400–1405, 2006.

[42] Patrick Pantel and Marco Pennacchiotti. Espresso: Leveraging Generic Patterns for Automatically Harvesting Semantic Relations. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, pages 113–120, 2006.

[43] Patrick Pantel and Deepak Ravichandran. Automatically Labeling Semantic Classes. In *Proceedings of Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT/NAACL-04)*, pages 321–328, 2004.

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei jing Zhu. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[45] Thanh Phong Pham, Hwee Tou Ng, and Wee Sun Lee. Word Sense Disambiguation with Semi-Supervised Learning. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1093–1098, 2005.

[46] Ellen Riloff. Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1044–1049, 1996.

[47] Ellen Riloff and Rosie Jones. Learning Dictionaries for Information Extraction by Multi-Level Bootstrapping. In *Proceedings of the Sixteenth National Conference on Artificial Intellligence (AAAI-99)*, pages 474–479, 1999.

[48] Eleanor Rosch. Cognitive Representation of Semantic Categories. *Journal of Experimental Psychology*, 104:192–233, 1975.

[49] Marco Saerens, François Fouss, Luh Yen, and Pierre Dupont. The principal component analysis of a graph, and its relationship to spectral clustering. In *Proceedings of European Conference on Machine Learning (ECML 2004)*, pages 371–383. Springer, 2004.

[50] Hinrich Schütze and Michael Walsh. A Graph-theoretic Model of Lexical Syntactic Acquisition. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 916–925, 2008.

[51] Satoshi Sekine and Hisami Suzuki. Acquiring Ontological Knowledge from Query Logs. In *Proceedings of the 16th international conference on World Wide Web*, pages 1223–1224, 2007.

[52] Craig Silverstein, Monika Henzinger, Hannes Marais, and Michael Moricz. *Analysis of a Very Large AltaVista Query Log*. Digital SRC Technical Note 1998-014, 1998.

[53] Alex J. Smola and Risi Imre Kondor. Kernels and Regularization of Graphs. In *Proceedings of the 16th Annual Conference on Learning Theory*, pages 144–158, 2003.

[54] Beth M. Sundheim. Overview of the fourth message understanding evaluation and conference. In *Proceedings of the 4th conference on Message understanding*, pages 3–21, 1992.

[55] Partha Pratim Talukdar, Joseph Reisinger, Marius Paşca, Deepak Ravichandran, Rahul Bhagat, and Fernando Pereira. Weakly-Supervised Acquisition of Labeled Class Instances using Graph Random Walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 581–589, 2008.

[56] Michael Thelen and Ellen Riloff. A Bootstrapping Method for Learning Semantic Lexicons using Extraction Pattern Contexts. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, pages 214–221, 2002.

[57] Jean Véronis. HyperLex: Lexical Cartography for Information Retrieval. *Computer Speech & Language*, 18(3):223–252, 2004.

[58] Ellen M. Voorhees. Overview of the TREC 2001 Question Answering Track. In *Text REtrieval Conference*, pages 71–79, 2001.

[59] Vishnu Vyas, Patrick Pantel, and Eric Crestan. Helping Editors Choose Better Seed Sets for Entity Set Expansion. In *Proceedings of ACM Conference on Information and Knowledge Management (CIKM-2009)*, pages 225–234, 2009.

[60] Gu Xu, Shuanghong Yang, and Hang Li. Named Entity Mining from Click-Through Log Using Weakly Supervised Latent Dirichlet Allocation. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1365–1373, 2009.

[61] David Yarowsky. Unsupervised Word Sense Disambiguation Rivaling Supervised Methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[62] Denyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Schökopf. Learning with Local and Global Consistency. *Advances in Neural Information Processing Systems*, 16:321–328, 2004.

List of Publications

Journal Papers

- 小町守, 飯田龍 (東工大), 乾健太郎, 松本裕治. 名詞句の語彙統語パターンを用いた事態性名詞の項構造解析. 自然言語処理, Vol.17, No.1, pp.141-159, January 2010.

- 小町守, 工藤拓 (Google), 新保仁, 松本裕治. Espresso 型ブートストラッピング法における意味ドリフトのグラフ理論に基づく分析. 人工知能学会論文誌, Vol.25, No.2, pp.233-242, January 2010.

- 小町守, 牧本慎平 (Yahoo Japan), 内海慶 (Yahoo Japan), 颯々野学 (Yahoo Japan). ラプラシアンラベル伝播による検索クリックスルーログからの意味カテゴリ獲得. 人工知能学会論文誌, Vol.25, No.1, pp.196-205, January 2010.

- 小町守, 鈴木久美 (Microsoft Research). 検索ログからの半教師あり意味知識獲得の改善. 人工知能学会論文誌, Vol.23, No.3, pp.217-225, March 2008.

International Confences (refereed)

- Mamoru Komachi, Shimpei Makimoto (Yahoo Japan), Kei Uchiumi (Yahoo Japan), Manabu Sassano (Yahoo Japan). Learning Semantic Categories from Clickthrough Logs. In Proceedings of the Joint conference of the 47th Annual Meeting of the Association for Computational

Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP 2009): Short Papers, pp.189-192. Singapore, August 2009.

- Mamoru Komachi, Taku Kudo (Google), Masashi Shimbo and Yuji Matsumoto. Graph-based Analysis of Semantic Drift in Espresso-like Bootstrapping Algorithms. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2008), pp.1011-1020. Honolulu, USA, October 2008.
- Mamoru Komachi and Hisami Suzuki (Microsoft Research). Minimally Supervised Learning of Semantic Knowledge from Query Logs. Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP-08), pp.358-365. Hyderabad, India, January 2008.
- Mamoru Komachi, Ryu Iida, Kentaro Inui and Yuji Matsumoto. Learning Based Argument Structure Analysis of Event-nouns in Japanese. Proceedings of the Conference of the Pacific Association for Computational Linguistics (PACLING), pp.120-128. Melbourne, Australia, September 2007.

International Workshops

- Mamoru Komachi, Masaaki Nagata (NTT) and Yuji Matsumoto. NAIST-NTT System Description for Patent Translation Task at NTCIR-7 In Proceedings of the NTCIR-7 Workshop, pp.435-440. Tokyo, Japan, Decemver 2008.
- Masaki Noguchi (Tokyo Institute of Technology), Kenta Miyoshi (Tokyo Institute of Technology), Takenobu Tokunaga (Tokyo Institute of Technology), Ryu Iida, Mamoru Komachi and Kentaro Inui. Multiple Purpose Annotation using SLAT - Segment and Link-based Annotation Tool -. In Proceedings of the 2nd Linguistic Annotation Workshop, pp.61-64, May 2008.
- Ryu Iida, Mamoru Komachi, Kentaro Inui, Yuji Matsumoto. Annotating a Japanese Text Corpus with Predicate-Argument and Coreference Relations. Proceedings of the Linguistic Annotation Workshop, pp.132-139. Prague, Czech, June 2007.

- Mamoru Komachi, Masaaki Nagata and Yuji Matsumoto. Phrase Reordering for Statisitcal Machine Translation Based on Predicate-Argument Structure. Proceedings of the International Workshop on Spoken Language Translation, pp.77-82. Kyoto, Japan, November 2006.

Book Chapters

- 「確率的言語モデルに基づくかな漢字変換」. 言語処理学事典. 共立出版. p33. December 2009.

Awards

- 情報処理学会第 191 回自然言語処理研究会第 74 回音声言語情報処理研究会学生奨励賞. 「ウェブ検索ログを用いたラベル伝播による意味カテゴリ獲得」"Learning Semantic Categories from Web Search Logs Using Label Propagation"

- 言語処理学会第 14 回年次大会最優秀発表賞. 「カーネル法を用いた意味的類似度の定義とブートストラップの一般化」"Kernel-based Definition of Semantic Similarity and Generalization of Bootstrapping"

- 言語処理学会第 13 回年次大会優秀発表賞. 「日本書き言葉を対象とした述語項構造と共参照関係のアノテーション: NAIST テキストコーパス開発の経験から」"Annotating Predicate-argument and Coreference Relations in Japanese Written Text: From the Experience of Building the NAIST Text Corpus"

Other Publications (domestic)

- 小町守, 牧本慎平 (ヤフー), 内海慶 (ヤフー), 颯々野学 (ヤフー). ウェブ検索ログを用いたラベル伝播による意味カテゴリ獲得. 情報処理学会第 191 回自然言語処理研究会. Vol.2009-NL-191, No.9, May 2009.

- 牧本慎平 (ヤフー), 小町守, 颯々野学 (ヤフー). ブートストラップを用いた検索クエリログからの意味カテゴリ獲得の分析 言語処理学会第 15 回年次大会論文集. March 2009.

- 小町守, 森信介 (京大), 徳永拓之 (PFI). 統計的かな漢字変換 ChaIME. 第 1 回入力メソッドワークショップ (WIM 2008). December 2008.

- 小町守, 森信介 (京大), 徳永拓之 (PFI). ChaIME: 大規模コーパスを用いた統計的かな漢字変換. 第 3 回 NLP 若手の会シンポジウム. September 2008.

- 小町守, 森信介 (京大), 徳永拓之 (PFI). あいまいな日本語のかな漢字変換. 情報処理学会夏のプログラミング・シンポジウム, pp.51-55. September 2008.

- 小町守, 工藤拓 (Google), 新保仁, 松本裕治. カーネル法を用いた意味的類似度の定義とブートストラップの一般化. 言語処理学会第 14 回年次大会論文集, pp.825-828. March 2008.

- 飯田龍, 小町守, 乾健太郎, 松本裕治. 名詞化された事態表現への意味的注釈付け. 言語処理学会第 14 回年次大会論文集, pp.277-280. March 2008.

- 野口正樹 (東工大), 三好健太 (東工大), 徳永健伸 (東工大), 飯田龍, 小町守, 乾健太郎. 汎用アノテーションツール SLAT. 言語処理学会第 14 回年次大会論文集, pp.269-272. March 2008.

- 小町守, 鈴木久美 (Microsoft Research). 検索ログを用いた意味知識獲得のためのブートストラップ手法. 電子情報通信学会言語理解とコミュニケーション研究会 (NLC 2007-10), 信学技報 Vol. 107, No. 246, pp.13-18. October 2007.

- 小町守, 飯田龍, 乾健太郎, 松本裕治. 事態性名詞の項構造解析における共起尺度と構文パターンの有効性の分析. 言語処理学会第 13 回年次大会論文集, pp.47-50, March 2007.

- 飯田龍, 小町守, 乾健太郎, 松本裕治. 日本語書き言葉を対象とした述語項構造と共参照関係のアノテーション: NAIST テキストコーパス開発の経験から. 言語処理学会第 13 回年次大会論文集, pp.282-285, March 2007.

- 飯田龍, 小町守, 乾健太郎, 松本裕治. NAIST テキストコーパス: 述語項構造と共参照関係のアノテーション. 情報処理学会自然言語処理研究会予稿集・NL-177-10・, pp.71-78, January 2007.

- 野口正樹 (東工大), 三好健太 (東工大), 徳永健伸 (東工大), 飯田龍, 小町守, 乾健太郎. セグメンテーションとリンクに基づくアノテーションツールの設計と実装. 言語処理学会第 13 回年次大会発表論文集, pp278-281, March 2007.

- 小町守, 飯田龍, 乾健太郎, 松本裕治. 共起用例と名詞の出現パターンを用いた動作性名詞の項構造解析. 言語処理学会第 12 回年次大会論文集, pp.821-824, March 2006.