# Data Mining Algorithms for Decision Support Based on User Activities

by

Ebad Ahmadzadeh

A dissertation
submitted to Florida Institute of Technology
in partial fulfillment of the requirements
for the degree of

Doctorate of Philosophy
in
Computer Science

Melbourne, Florida
July, 2018

We the undersigned committee
hereby approve the attached dissertation

**Data Mining Algorithms for Decision Support Based on User Activities** by Ebad Ahmadzadeh

---

Philip K. Chan, Ph.D.
Associate Professor
School of Computing
Committee Chair

---

Georgios Anagnostopoulos, Ph.D.
Associate Professor
Electrical and Computer Engineering
Outside Committee Member

---

Debasis Mitra, Ph.D.
Professor
School of Computing
Committee Member

---

Heather Crawford, Ph.D.
Assistant Professor
School of Computing
Committee Member

---

Phil Bernhard, Ph.D.
Associate Professor and Director
School of Computing

ABSTRACT

Title:

**Data Mining Algorithms for Decision Support Based on User Activities**

Author:

Ebad Ahmadzadeh

Major Advisor:

Philip K. Chan, Ph.D.

This dissertation covers four data mining problems with applications in decision support based on user activity data. The first problem is an efficient approach to maximizing spread of information in social networks with applications in decision support for marketing where the goal is to find the best set of users, based on a limited budget, to maximize the word of mouth. The data for this problem is based on user activities in social networks that lead to formation of friendship (or follower-followee) graphs. The second problem is identifying action-outcome relationships to facilitate building a knowledge base of actions that could be used for decision support. The data for this problem is based on user experience about performing actions as expressed on social media. The third problem is automatic extraction of relevant product aspects in a summarized form as well as a list of pros and cons for each aspect. Identifying strengths and weaknesses of a product can be useful in the decision making process for the company that makes the product to improve the weaknesses and add desired features. We use real wold data sets based on user activities from social media to evaluate our proposed techniques. The fourth problem provides access control decision support for smartphone devices by distinguishing between device owner and others based on their typing patterns and

device movements.

# Table of Contents

[1]https://nlp.stanford.edu/projects/glove/

# List of Figures

# List of Tables

# List of Symbols, Nomenclature or Abbreviations

| | |
|---|---|
| SS | Significance Score (Chapter 4) |
| SS2 | Significance Score 2 (Chapter 5) |
| SSS | Steady State Spread (Chapter 3) |
| eSSS | Estimated Steady State Spread (Chapter 3) |
| VSM | Vector-based Spread Maximization (Chapter 3) |
| AEMI | Augmented Expected Mutual Information (Chapter 5) |
| AUC | Area Under ROC Curve (Chapter 6) |
| DCG | Discounted Cumulative Gain (Chapter 4,5) |
| POS | Part-of-Speech (Chapter 4) |
| LDA | Latent Dirichlet Allocation (Chapter 5) |

# Acknowledgements

I would like to thank my academic advisor, Dr. Chan for his dedication, support, encouragement and patience throughout my Ph.D. program. I would like to thank Dr. Crawford, as my initial advisor and committee member for her guidance and support during my doctorate program since 2013. Also, a special thanks to Dr. Mitra, who played an important role in my education both as a professor and a committee member. His insights and views have always inspired me. Also, I would like to thank Dr. Anagnostopoulos as my committee member for his continuous support and encouragement. Finally, I would like to thank Accent Technologies, Inc., specially Mr. Pete Mcchrystal for his strong support, patience and belief in this research. Parts of the current research for the period between Aug 2016 till May 2018 was supported by this company as a research grant.

# Dedication

This dissertation work is dedicated to my wife, Mary, for her encouragement and support during many challenges in our journey. Her genuine and convivial spirit has been the reason for our happy and successful life together. This work is also dedicated to my parents who taught me to be loving, patient and independent.

# Chapter 1

# Introduction

Today, many questions can be answered to assist in decision making process, given the large volume of data available online in different forms like wiki, social media, user reviews, etc. Data mining algorithms are applied to such problems and provide decision support for individuals, companies and even other computers. This dissertation covers four data mining problems with applications in decision support based on user activity data. Each problem addresses a real-world challenge in decision support, and the nature of data is always based on user activities online or on mobile devices. We provide an overview about each problem, as well as our proposed solution and the nature of data for each problem. We also explain how each solution helps facilitate the decision process.

## 1.1 Problem Overview

- **Problem 1:** Influence maximization has been studied in different areas such as social networks and marketing. Given a graph, the task is to find a small set of nodes whose aggregated influence is maximum on the entire network. In social networks, these nodes are referred to as influential people [73]. A famous decision support application of this problem is in marketing where the goal is to decide whom to market first to influence others, so that the spread of influence is maximized. Specifically, the goal is to find a set of limited users, based on the available budget, to maximize the influence (spread of the marketing message) on the network. The data for this problem is based on user activities in social networks that lead to formation of friendship (or follower-followee) graphs. Influential users tend to be connected more users where the direction of communication is mostly outwards.

- **Problem 2:** Identifying action-outcome relationships could facilitate building a knowledge base of actions that could be used for decision support. The main question is that what consequences one should expect by performing a certain action. These likely consequences are obtained from experiences of others on social media that performed the same action and reported the outcomes. The main task is to identify likely consequences of doing actions in form of pros and cons based on the experience of others mentioned on social media. The input includes a large corpus of personal status messages from social media (e.g. Twitter), and a query specifying an action or a goal depending on the question type. The output consists of a list of consequences of doing an action arranged in a list of pros and cons, or a list of actions that lead to achievement of the given goal. For example, the question can be to find the most likely consequences of adopting a cat, in which case the query is "adopting a cat". Then, the output is a list of consequences (pros and cons) of adopting a cat.

- **Problem 3:** In recent years, the volume of user-generated social media content has been growing. People across the Web are constantly sharing experiences and opinions about a wide range of situations. Many research lines have focused on using this information as a data source to apply to domains like decision support. Reviews of products and services is one of the highly valuable sources of data that can be used to answer interesting questions about the product or service. For example, identifying strengths and weaknesses of a product can be useful for the company that makes the product to improve the weaknesses and add desired features. Given a collection of product reviews, the goal is to automatically extract relevant product aspects and to find the most significant sentences that represent pros and cons for each aspect. The input is a corpus of product reviews, and the output is product aspects that are often discussed in the reviews, as well as a list of pros and cons for each of the aspects. For example if the product is a cell phone, then aspects could be call quality, price, camera, etc.

- **Problem 4:** Smartphones and other wearable devices have become well-equipped with various sensors such as gyroscope, GPS, etc. As those various data types are available, novel problems can be solved. For example, device owners can be verified based on their usage patterns extracted from the sensors. In this chapter, we aim to use data mining algorithms to model user behavior based on such data sources. The main goal is to provide a decision support solution for smartphones to distinguish between the device owner and others, and to make access control decisions. The access control decisions can allow or disallow user access to an app or some data. We use user activity data based on typing patters and device movements to model user behavior.

## 1.2 Approach Overview

- **Problem 1:** We propose a Vector-based Spread Maximization (VSM) algorithm that leverages spread information in the initial SSS (Steady State Spread) calls and estimates the SSS value for future calls to reduce computation. We also improve the efficiency of SSS by restricting its search space. The original SSS method updates all nodes except the seeds, however, many of them will remain zero, particularly in a large sparse graph, because they are not reachable. To improve the efficiency of SSS for large sparse graphs, we only update nodes that will have positive spread.

- **Problem 2:** we mine related pros and cons of the action via extracting significant events as potential outcomes of the action. We propose using actions and characteristics to select relevant messages, and adjective vectors to establish similarity among adjectives. We introduce SS (Significance Score) to select event headlines, and to rank them in the final pros-and-cons table.

- **Problem 3:** Our approach has three main steps. After preprocessing the review text (e.g. removing URLs, emojis and bad characters) and tokenizing them into sentences, the first step is to find the best set of product aspects and assign the review sentences based on the most probable aspects. Second, we employ an scoring method to select messages that are likely to represent pros or cons. We use this score to identify top $K$ sentences for each topic. Then, we use sentiment intensify score with a minimum threshold to separate pros from cons. Third, we summarize product aspects from each topic and present them as bigram phrases using ordered AEMI (Augmented Expected Mutual Information) [77].

- **Problem 4:** Our approach involves a two-stage classification. In the first stage, we determine user position type (SIT, STAND, or WALK). Then, in the second stage, we verify the user based on the keystroke and bigram data collected from their typing patterns. We show that our two-stage method provides high AUC (Area Under ROC Curve) based on a user study on 30 users.

## 1.3 Key Contributions

- **Problem 1:** Our main contribution on this problem is an efficient method to estimate SSS of multiple seeds from SSS of individual seeds. Plus, our VSM generally is slightly more effective than existing algorithms and significantly more efficient.

- **Problem 2:** Our contribution on this problem is identifying relevant messages by extracting actions and characteristics. Also, we introduce Adjective Vectors to measure semantic similarity between adjectives to improve the clustering quality. Additionally, we propose a Significance Score (SS) to quantify significance of messages in terms of representing meaningful outcomes.

- **Problem 3:** We jointly identify product aspects along with pros and cons with respect to each aspect. Also, we summarize the aspects in form of bigrams that show different descriptions or opinions about each product aspect. Plus, we propose a modified Significant Score (SS2) with additional factors to quantify significance of sentences in terms of representing meaningful pros or cons with respect to the product aspects.

- **Problem 4:** Our key contribution in this problem is the two-phase classification technique where we create three different models based on user positions (SIT, WALK, STAND). The results show that our two-phase classification approach achieves higher AUC (more than 91%) and lower FRR and FAR (less than 7%) compared to a baseline (1-phase) method in distinguishing between device owner and others.

## 1.4 Organization

In Chapter 2 we provide an overview of the related work to the data mining problems in decision support, influence maximization on social media, actions and outcomes based on social media, summarization of product aspects, and identifying pros and cons for product aspects.

In Chapter 3 we provide a solution with improved efficiency for the problem of maximizing the influence spread on social media.

In Chapter 4 we provide a solution for a the problem of identifying pros and cons of actions based on user experience expressed in social media.

In Chapter 5 we propose solutions for the problem of extracting product aspects and providing aspect summaries as well as finding pros and cons for each aspect.

In Chapter 6 we propose a solution for access control decision support on smartphones by verifying device owners based on the patterns extracted from their typing patterns and device movements.

Finally, we conclude this dissertation by presenting a summary of our approaches and contributions in Chapter 7

# Chapter 2

# Related Work

## 2.1 Problem 1: Improving Efficiency of Maximizing Spread in the Flow Authority Model for Large Sparse Networks

Given a graph, Kempe et al. [73] discuss two diffusion models: Independent Cascade (IC) and Linear Threshold (LT). In the IC model, each node has only one chance to influence its neighbors. In the LT model, each node has an activation threshold; a node is active when the total influence from its active neighbors exceeds the threshold. Aggarwal et al. [4] introduce the Flow Authority model. Different from the two models above, the expected number of active nodes is directly estimated, instead of running (e.g. 10,000 [73]) Monte Carlo simulations, which could be computationally expensive. The key question of these three models is how to find the initial seed set that maximizes the expected number of active nodes.

The methods used in related work can be categorized into two general approaches. The first general approach uses an evaluation function for a node or a set of nodes to find the top set of nodes. Different sets of nodes are generated and the evaluation function guides the selection. For example, the Greedy [73] algorithm starts with sets that contain only one node, each successively generates sets with one additional node, and selects the set that maximizes the evaluation function, which is the objective function for the problem. Kimura et al. [80] use the Greedy algorithm and bond percolation as the evaluation function. Based on the submodularity property of the objective function, CELF [88] improves the Greedy algorithm with a "lazy forward" evaluation technique, which prunes nodes that cannot improve the set. Degree Discount [28] is similar except the evaluation function is a heuristic based on the neighbors of the node. PMIA [27] uses a tree of nodes with maximum influence to construct a heuristic as the evaluation function. CGA [131] first identifies communities in the graph and then greedily selects nodes from the communities. SIMPATH [53] estimates the spread from a set by exploring paths from the set up to a threshold as the evaluation function. Jiang et al. [70] initially select a set of random seeds then use simulated annealing to evaluate neighboring seed sets by replacing one of the seeds. Independent Path Algorithm (IPA) [78] evaluates each candidate by calculating the spread from the current set and the candidate to the descendants of the candidate. They store influence paths for each node, but they limit the number of nodes to reduce memory usage. Borgs et al. [20] propose a nearly-optimal-time algorithm for the IC model that chooses a random set of initial nodes and finds their ancestors, which are seed candidates. The evaluation function of a seed candidate is the number of times it is an ancestor. Recognizing the large constant in the time complexity, TIM+

[122] improves Borgs et al.' algorithm by bounding the number of initial nodes to a smaller number. IMM [121] further reduces the number of initial nodes by a Martingale approach, which allows some dependency between successive runs of finding ancestors to determine the number of initial nodes. Based on the Flow Authority model, Aggarwal et al. [4] propose Steady State Spread (SSS) as the objective and evaluation functions for a node in their algorithms. RankedReplace selects individual nodes with the highest SSS as the initial set and successively attempts to replace one in the set by one outside the set. Using the DBLP data set, they illustrated that the top set of authors found by their algorithms are more recognizable than sets found by two other algorithms a heuristic of Degree Discount [29] and Peer-Influence [40].

In the first general approach, many algorithms greedily add a node to the set and do not attempt to change previously added nodes. The exceptions are the simulated annealing approach [70] and RankedReplace[4]. Both approaches iteratively evaluate the neighboring seed sets, by replacing a node in the seed set with another not in the seed set. If the replacement improves the objective function, both approaches keep the replacement. However, simulated annealing keeps a replacement that does not improve with some probability. In simulated annealing, the initial seed set is randomly selected. However, RankedReplace creates the initial set by selecting nodes based on their individual values of the objective function in the Rank step. Since RankedReplace chooses the initial seeds independently without considering their interactions, more replacements might be needed in the Replace step. Also, each attempt of replacement invokes the objective function SSS, which could be expensive.

The second general approach propagates values according to the graph structure and selects the top nodes with the highest values. Bayes Traceback [4] starts with equal probability for each node and back propagates probabilities to its in-neighbors. At each iteration, a fraction of the nodes with the lowest probabilities are removed and the probabilities of the remaining nodes are redistributed.

## 2.2 Problem 2: Mining Pros and Cons of Actions from Social Media for Decision Support

Much research exist based on the assumption that co-occurrence may establish some true relationships between actions and outcomes. For instance, in the health domain, social media studies have found relationships among diseases, medicines, related symptoms and side-effects [103].

Richardson [113] uses search queries to identify relationships between drugs and their adverse side-effects (consequences). Similar studies address the problem of learning about the real world events from social media. They predict the future signals from social media given a known signal. These techniques are applied to different domains like economics [19]

Olteanu et al. [106] performed an open-domain study on words expressed by social media users after experiencing distinct situations, and found that causal relationships between those words and the situations are in average 55-100% more likely than semantic relationships.

Similarity between words can also be measured by vector representation of words. Training of such vectors has been done via different techniques like the ones based on matrix factorization [39] and window-based methods [99]. Pennington et al. [112] propose GloVe, an unsupervised method that benefits from both families and outperforms them on word similarity, word analogy, and name entity recognition tasks.

Kiciman and Richardson [74] investigate the feasibility of mining the relationship between actions and their consequences based on social media. The inputs include a large corpus of personal status messages from social media and an action query. The output is a list of pros and cons of doing the action. A timeline of events is constructed for each user, where each event is a collection of relevant personal-experience posts. The user timelines are center-aligned at the point of performing the action. They are then divided into two groups of positive and negative user timelines representing those who did the action and others who did the reverse action respectively. Finally, most important events are found based on relative likelihood, and they are split into pros and cons via aggregate affect valence.

One of the main shortcomings of [74] is in the event extraction step where sentences are broken into phrases and then clustered into events. Events consist of short phrases that could be less meaningful sometimes. For example, "damn kitten" or "cat is literally" are phrases from their output table that could not express an outcome without referring to the message they belong to. Therefore, selecting messages that represent the event-phrase seems to be important. However, how to pick the example messages in the pros-and-cons is not clear. Furthermore, they performed semantic correlational analysis to order the events with respect to relative likelihood of the event occurring after doing the action compared to both before doing the action and after doing the reverse action. Although the relative likelihood score captures distinguishing events, the results potentially contain events that are not important consequences. For example, "cat being named" is in the results, but it doesn't seem to be the most significant outcome of adopting a cat.

## 2.3 Problem 3: Identifying Pros and Cons of Product Aspects Based on Customer Reviews

The task of identifying pros and cons from product reviews usually involves three procedures; identifying product aspects, sentiment analysis, and summarization. We briefly review some existing algorithms for each task as they relate to our task.

The most popular techniques for identifying product aspects are based on frequency analysis [64, 60, 91]. Zhao et al. [137] propose a technique based on syntactic structures. Other studies have explored supervised [68] and unsupervised [85] techniques. Aspect-based sentiment analysis can be done as a joint task where the goal is to calculate sentiment score for each product aspect [104, 83].

Sentiment analysis can be done at different levels such as word, phrase, sentence and document. VADER [47] is an unsupervised, lexicon and rule-based method tuned for sentiments of words, phrases and sentences expressed in social media. Among the state of the art supervised techniques those of Socher et al. [118] can be mentioned. Moreover, document-level sentiment classification techniques, often applied to reviews, have been explored extensively [107, 127]. The task of distinguishing between subjective and objective expressions is useful to separate opinions from facts [134, 115] with applications in question answering, summarization, etc. Subjective clues were collected as part of the work reported in [114].

The main goal of summarization task is to generate a short but meaningful representation given a larger text. Hu and Lie [64] use feature-opinion pairs to summarize reviews. Lu et al. [92] have proposed techniques based on clustering of phrases and aspects.

As discussed above, many of the existing research focus on one or two of the tasks but not all three. For instance, Zhao et al. [137] focus on aspect extraction, Hai et al. [61] propose a supervised technique for joint modeling of aspects and sentiment, but they do not provide a solution to summarize the reviews in form of pros and cons. Ahmadzadeh and Chan [5] propose a method to identify pros and cons of doing actions based on social media. However, they do not extract aspects. Also, their solution is based on events before and after doing the action (e.g. purchasing the product in this case). But product reviews usually do not include much information about user experience before purchasing the product.

To the best of our knowledge the closest work to ours is that of Kim and Hovy [79], a supervised method for identifying pros and cons from product reviews. They use three categories of features to train; 1) Lexical Features consist of unigrams, bigrams and trigrams that represent reasoning tokens like “that’s why”, 2) Positional Features specify whether the sentence is from the first, second, last or second to the last sentence of the review paragraph, 3) Opinion-bearing Word Features consist of a dictionary of pre-selected opinion words. Each learning instance is a sentence associated with a label (“pro”, “con” or “neither”). They separate the task of finding pro and con sentences into two phases each being a binary classification. In the first phase (identification), they separate “pro” and “con” sentences from “neither”, and in the second phase (classification), they classify the candidates into pros and cons. Still they do not provide a method to extract aspects. Therefore, the pros and cons identified by their algorithm are at the product-level. Whereas, in our work we identify pros and cons at the aspect-level. For example, given a coffee maker as the product, our algorithm first finds aspects like cup size and water reservoir, and then identifies pros and cons for each aspect.

## 2.4 Problem 4: Authentication on the Go: Assessing the Effect of Movement on Mobile Device Keystroke Dynamics

### 2.4.1 Mobile Device Authentication

In addition to existing password- and PIN-based authentication methods, research has begun to emerge on alternative authentication methods that consider the mobile device’s needs more closely; in particular interest in using graphical passwords as an authenticator has been demonstrated [41, 120]. However, these methods still provide an all-or-nothing approach to device protection in that once the user is correctly authenticated, they are granted access to all data, services and apps on the device. In response, researchers have begun to study methods that continue to authenticate the user invisibly in the background while other tasks are completed. This is called *transparent, continuous* authentication. This type of authentication gathers behavioral biometric data such as keystroke dynamics [22, 94], touches [135], etc. to continuously ensure that the device owner is the one currently using the device. Methods by which access to apps, data and services on the device can be restricted based on the identity of the current user have also begun to emerge [37].

### 2.4.2 Keystroke Dynamics

Keystroke dynamics is a behavioral biometric that uses patterns in how a person types to distinguish them from other users. It uses metrics such as *key hold time* (the amount of time between pressing and releasing the same key) and *inter-key latency* (the amount of time that passes between releasing one key and pressing the next) to identify these distinctive patterns. Researchers have examined other potential biometrics such as touch [24], facial recognition [33] and device movement [35]. Keystroke dynamics began with studies on desktop and laptop computers [86, 102] and in recent years has moved to mobile devices such as smartphones [32, 94]. Many keystroke dynamics studies attempt to replicate a password hardening situation in which the data gathered during the study is based on a known password that each study participant types a set number of times [76]. This practice can increase the strength of traditional passwords, but still provides an all-or-nothing approach to authentication. More recently, research has focused on providing transparent authentication that protects throughout device use rather than just at the beginning. It is this research that maps most directly to ours; thus we will focus on the current work in this area.

Typing patterns while moving have been studied by Clawson *et al.* in an effort to determine whether moving while typing affects accuracy and errors [34]. Their study had 36 participants type set phrases using a hard keyboard on a mobile device (Blackberry Curve 8320) while walking a set path. Their results showed that expert typists made fewer mistakes while walking, but at the cost of a lower typing speed [34]. Clawson *et al.*'s work is supportive of ours since more accurate typing may lead to improved uniqueness in typing patterns. However, Clawson *et al.* were not studying the use of keystroke dynamics as an authentication method, so further comparison of our results to this work are not indicated.

There has been much discussion on the amount and type of text used as input to transparent keystroke dynamics authentication tools. Many of the studies in this area, specifically those to do with password hardening, focus on text that must be repeated, while continuous, transparent authentication methods are likely to be based on any text the user may type. There is also the need for ecological validity – if a user can be expected to type any words and phrases, then basing a user study on specific words or phrases cannot be used to justify results in a more open environment.

#### 2.4.2.1 Fixed Text

Fixed text methods (also called *static text*) assume that the user will type the same word or phrase at both enrollment and at the time of authentication. The text typed is generally short, as typing long texts at the time of authentication is tedious and error-prone on a mobile device [7, 26]. In general, using fixed text allows for more stability as the comparison between enrolled sample and gathered sample share the same keys and are thus similar. In some cases, experiments of this type produce results that either depend on special conditions (such as the attacker knowing the user's password) or have unacceptable accuracy levels [55, 17]. Much research has been done on fixed text methods [31, 76]; a summary of work in this area may be found in [36].

#### 2.4.2.2 Dynamic Text

Also called *free text*, this paradigm assumes that the user may type whatever they wish, and that this input is any length. In reality, dynamic text and free text have several differences; free text is completely without constraints, where dynamic text may have aspects of both fixed and free text. Specifically, dynamic text may be prompted in some way, or may depend on a small number of specific words or phrases [123]. Several studies have examined dynamic text keystroke dynamics [9, 98], including Ahmed *et al.* [6] who report fairly good results, with False Accept Rates well below 1%. Free text keystroke dynamics has also been studied by Gunetti & Picardi [55], although their reported results are not as low as those of Ahmed *et al.* A summary of work on free text keystroke dynamics is available in [8]. The implication is that transparent authentication based on keystroke dynamics is best suited to *true* free text, which removes any restrictions about what or how much is typed. In this way, any characters a user may type can potentially be used as information upon which to base authentication decisions.

Adding realism to mobile device keystroke dynamics experiments has been studied from several points of view. One is that users may change their hand positioning while typing, which may affect their overall typing pattern. Azenkot & Zhai [13] studied user typing patterns when typing with one thumb, both thumbs and one index finger and found that there were pattern differences between these three hand positions. They used these results to suggest changes in keyboard design and layout that can improve typing accuracy. Similarly, Buschek *et al.* [23] studied the same three hand positions, but from the point of view of authentication rather than keyboard improvements. Their results showed that hand position had a strong effect on the ability to authenticate a user [23]. Both of these papers were based on password-hardening techniques, and thus were using fixed text techniques with defined feature vectors. Shen *et al.* studied the use of motion sensor data while typing a mobile device passcode as a potential authenticator [116]. They reported a False Reject Rate (FRR) of 6.85% and a False Accept Rate (FAR) of 5.01% in a user study with 48 participants [116]. Their work is similar to ours since they report results in the seated, standing and walking positions, although they only consider the sensor data when unlocking the mobile device with a passcode.

### 2.4.3 Gyroscope Data

Modern mobile devices come equipped with built-in sensors that can measure motion, orientation, environmental conditions such as temperature and humidity, etc. Data from sensors such as accelerometers and gyroscopes has been used for activity recognition [21, 57], to address typing inaccuracies [49], to create keyloggers [96], and to determine on-device input errors [105]. Accordingly, authentication research has begun to consider whether accelerometer and gyroscope data may be used as a unique identifier. Giuffrida *et al.* created what they call "sensor-enhanced" keystroke dynamics in their UNAGI system [48]. They experimented with the use of accelerometer and gyroscope data while 20 participants typed a set of fixed passwords and found that they were able to achieve Equal Error Rate (EER) values of less than 1% [48]. Their use of a fixed password as the stimulus indicates that theirs was a password hardening experiment rather than a dynamic text experiment.

# Chapter 3

# Improving Efficiency of Maximizing Spread in the Flow Authority Model for Large Sparse Networks

## 3.1 Introduction

Given a graph, a number of researchers in different areas have studied how to find the set of most "important" nodes in the graph. In social networks, important nodes are influential people [73]. For marketing purposes, the problem is to decide whom to market first, who in turn influence others, so that the spread of influence is maximized. In detecting water contaminants, important nodes are sensor locations in the water distribution network [88]. The problem is to identify a set of sensor locations that minimizes the total cost of detecting contaminants in the network. Other applications include finding a small set of “leaders” who could coordinate a network of distributed agents/robots. Aggarwal et al. [4] introduce the Flow Authority (FA) model, which specifies how information flows from nodes to their neighbors. Given a graph, the main question is how to efficiently find a set of nodes that initially has the information and maximizes the expected number of nodes that will assimilate the information. The authors define Steady-state Spread (SSS) as the objective function and propose RankedReplace as an algorithm to maximize SSS. RankedReplace repeatedly calls the objective function to guide its search for the top set of seeds, however, each SSS call can be time consuming.

We propose VSM that leverages spread information in the initial SSS calls and estimates the SSS value for future calls to reduce computation. Our main contributions include:

- an efficient method to estimate SSS of multiple seeds from SSS of individual seeds,
- our proposed VSM algorithm generally is slightly more effective ($< 1\%$) than existing algorithms and significantly more efficient than RankedReplace in 3 real-world datasets, and
- a significantly more efficient SSS algorithm.

We discuss related work in Chapter 2. Sec. 5.2 provides the problem statement and more background on the RankedReplace algorithm and the SSS function. Sec. 3.3 introduces our VSM algorithm. Sec. 3.4 discusses a more efficient SSS algorithm for large sparse graphs. We evaluate our algorithms in Sec. 3.5 and conclude in Sec. 3.6.

## 3.2 Problem Statement and Background

*Flow authorities* are the nodes that cause maximum spread of information in social networks. We use the same formulation as Aggarwal et al. [4]. Consider a directed network $G = (V, E)$, where $V$ is a set of nodes and $E$ is a set of edges. Moreover, each edge $e = (i, j)$ of the network is associated with a propagation probability $P_{ij}$ which specifies the probability by which the information propagated by node $i$ is absorbed at the destination node $j$. This is based on the assumption that if node $i$ has the information, all its neighbors are automatically exposed to the information, and the assimilation probability is $P_{ij}$ for each neighbor node $j$. Given a set $S$ of $k$ nodes, we define $\pi(i)$ to be the steady-state probability that node $i$ assimilates the information. The expected number of nodes, or Steady-State Spread (SSS), which assimilate the information is:$SSS(S) = \sum_{i \in V} \pi(i)$. The goal is to find $S$ of size $k$ such that $SSS(S)$ is maximized.

### 3.2.1 Steady-state Spread and RankedReplace

Aggarwal et al. [4] proposed RankedReplace to find $S$ such that the objective function $SSS(S)$ is maximized. They first calculate $\pi(i)$, which is the steady-state probability that node $i$ assimilates the information (is activated). The basic idea is that node $i$ is activated if it receives the information from at least one of its neighbors. Then, $\pi(i)$ is calculated as:

$$\pi(i) = 1 - \prod\nolimits_{l \in N(i)} (1 - \pi(l) p_{li}), \tag{3.1}$$

where $N(i)$ is the set of in-neighbors of node $i$, and $p_{li}$ is the propagation probability from node $l$ to $i$. Alg. 1 calculates *SSS*–the input includes an initial set $S$ and

**Algorithm 1** SSS($S$, $P$)

```
1: $\forall i \in S, \quad q^0(i) \leftarrow 1$
2: $\forall i \notin S, \quad q^0(i) \leftarrow 0$
3: $t \leftarrow 0$
4: **repeat**
5:     $\forall i \in S, \quad q^{t+1}(i) \leftarrow 1$
6:     $\forall i \notin S, \quad q^{t+1}(i) \leftarrow 1 - \prod_{j \in N(i)}(1 - P_{ji} \cdot q^t(l))$
7:     $C_{t+1} \leftarrow \sum_{i \notin S} |q^{t+1}(i) - q^t(i)|$
8:     $t \leftarrow t + 1$
9: **until** $C_t < 0.01 \cdot C_1$
10: **return** $\sum_{i \notin S} q^t(i)$
```

a propagation probability matrix $P$. $q^t(i)$ is the estimate of the steady state probability of node $i$ having the information at time $t$. Initially, the value of $q(i)$ is set to 1 where $i{\in}S$, and 0 where $i{\notin}S$. Then $q^t(i)$ is iteratively updated by calculating the probability that at least one of $i$'s neighbors spreads the information to $i$ (line 6) until the total spread converges. The Rank step in RankedReplace performs SSS for each node in the graph and the top $k$ nodes form the initial $S$. In the Replace step, a node in $S$ is replaced with a node in $V \backslash S$ if the SSS value improves. The algorithm stops if no replacement was made after $r$ trials.

## 3.3 Vector-based Spread Maximization

The Replace step in RankedReplace calls the expensive SSS (Alg. 1) for different seed sets. Also, the Rank step does not consider seed interactions to create the initial seed set, which could result in more replacements in the Replace step. Our VSM (Vector-based Spread Maximization) algorithm efficiently estimates SSS and considers seed interactions.

**Algorithm 2** Greedy($f, k$)

```
1: $S \leftarrow \emptyset$
2: for $i \leftarrow 1$ to $k$ do
3:     $u \leftarrow argmax_{c \in V \setminus S}(f(S \cup \{c\} - f(S))$        ▷ gain
4:     $S \leftarrow S \cup \{u\}$
5: return $S$
```

### 3.3.1 Greedy Algorithm

For the IC and LT models, Kempe et. al. [73] show that the problem of finding a seed set $S$ of size $k$ that maximizes the total spread is NP-Hard. However, they also prove that if the function $f$ is non-negative, monotone, and submodular, a general greedy approach, shown in Alg 2, guarantees a solution to be at least 1 - 1/$e$ (63%) of the optimal solution. It iteratively finds new seed nodes that yield the highest spread gain, and it stops when $k$ such seed nodes are found. Given the seed set *S*, spread function *f*, and a candidate node *c*, the gain is calculated as: $f(S \cup \{c\}) - f(S)$. SSS (Alg. 1) is an option for *f*, however, it is relatively computationally expensive. To improve efficiency, we propose estimating SSS without running Alg. 1 for seed sets with more than one seed.

### 3.3.2 Estimating SSS

To estimate SSS from multiple seeds, we store and use the last vector $q^t$ from SSS (Algorithm 1) with seed set of size 1. This vector, which we call SSS-vector, contains the influence spread value of the given seed set on every node in the graph. Consider $v$ is a seed, we denote the SSS-vector of $v$ as $q_v$ and $q_v(i)$ as the spread from $v$ to $i$. By storing the SSS-vectors, we can calculate the estimated spread, called eSSS, much faster (i.e. lower running time). For example, for a set of nodes $v_1, v_2, ..., v_n$, if we calculate and store the SSS-vectors for each of them, then in order to calculate $SSS(v_1, v_2)$ we do not need to run SSS. Instead, we could find the estimated SSS (eSSS) by aggregating the stored SSS-Vectors of $v_1$ and $v_2$. To aggregate the SSS-vectors (or "vectors" when the context is clear) from two seeds, we calculate the probability that the information is spread to $i$ from the first seed or the second seed. This probability is the probability complement of neither the first seed nor the second seed spread the information to $i$. Consider $q_x$ is the vector from seed $x$, $q_y$ is the vector from seed $y$, and $q_x(i)$ and $q_y(i)$ are the spread probabilities at node $i$ from the two seeds. We use $\oplus$ to denote the aggregate operator for two vectors:

$$\begin{aligned} q_{\{x,y\}} &= q_x \oplus q_y \\ q_{\{x,y\}}(i) &= 1 - (1 - q_x(i)) \times (1 - q_y(i)), \\ &= q_x(i) + q_y(i) - q_x(i)q_y(i), \end{aligned} \tag{3.2}$$

where vector $q_{\{x,y\}}$ is the result of aggregating vectors $q_x$ and $q_y$, and $q_{\{x,y\}}(i)$ is the aggregated spread to node $i$. The general case for aggregating vectors:

$$\begin{aligned} q_S &= q_{v_1} \oplus q_{v_2} \oplus ... \oplus q_{v_k} \\ q_S(i) &= 1 - \prod_j^k (1 - q_{v_j}(i)), \end{aligned} \tag{3.3}$$

where $S = \bigcup_j^k \{v_j\}$. The aggregate operator is "cumulative:"

$$q_{S \cup \{c\}} = q_S \oplus q_c$$

since:

$$\begin{aligned} q_{S\cup\{c\}}(i) &= 1 - \left(\prod_j^k (1 - q_{v_j}(i))\right)(1 - q_c(i)) \\ &= 1 - \left(1 - \left(1 - \prod_j^k (1 - q_{v_j}(i))\right)\right)(1 - q_c(i)) \\ &= 1 - (1 - q_S(i))(1 - q_c(i)) \end{aligned}$$

That is, we can aggregate the SSS-vector of candidate $c$ and the SSS-Vector of $S$ without using Eq.3.3. We note that Eq.3.2 (or similarly Eq.3.3) is only needed when $q_x(i)$ and $q_y(i)$ are both positive, which means that both $x$ and $y$ influence $i$. Otherwise, if one of them is zero, $q_{S\cup\{c\}}(i)$ is updated to be the non-zero value (which is mathematically equivalent to Eq.3.2). Similarly, if both $q_x(i)$ and $q_y(i)$ are zero, $q_{S\cup\{c\}}(i)$ is not updated. We next discuss techniques that improve the estimation.

#### 3.3.2.1 The Effect of Multiple Seeds on a Common Path

One source of error in the estimation of SSS is when multiple seeds influence a node via a common path. We show this issue with an example illustrated in Fig. 3.1(A) where two seeds $a$ and $b$ share a common path to influence node $y$. In this case, $SSS(\{a, b\})$ to node $y$ is $\alpha\gamma + \beta\gamma - \alpha\beta\gamma$. Because, according to Alg. 1, at the first iteration, $SSS(\{a, b\})$ is zero on $y$, but it is $\alpha + \beta - \alpha\beta$ on $x$. At the second iteration $SSS(\{a, b\})$ becomes $\alpha\gamma + \beta\gamma - \alpha\beta\gamma$ to $y$. However, the estimation, using Eq.3.2, would yield $\alpha\gamma + \beta\gamma - \alpha\beta\gamma^2$. Because $q_a(y) = \alpha\gamma$ and $q_b(y) = \beta\gamma$. This estimation could be corrected by dividing the third term by $\gamma$. Our second example shown in Fig. 3.1(B) depicts a longer common path from the two seeds to node $z$. In this case, similar to the previous one, $SSS(a, b)$ to node $z$ is $\alpha\gamma\lambda + \beta\gamma\lambda - \alpha\beta\gamma\lambda$.

However, the estimation using Eq.3.2 is $\alpha\gamma\lambda + \beta\gamma\lambda - \alpha\beta\gamma^2\lambda^2$. The weights $\gamma$ and $\lambda$ in the common path should be discounted from the third term. Generally, the estimation could be corrected by dividing the third term by the product of the weights on the common path.

Our last example illustrated in Fig. 3.1(C), shows a more complicated case where more than one common paths exist from the seeds to a node like $z$. However in this case, we cannot correct the estimation based on the vector entries and the common path weights, as the terms are not easy to decompose unless we store more information. So, we only consider to address situations like cases A and B. Another reason for this decision is that finding all such common paths can be computationally expensive. Calculating single common path seems to be an appropriate balance between efficiency and accuracy because it is easy to calculate and it improves the estimation accuracy. Hence, Eq. 3.2 is updated as:

$$q_{\{x,y\}}(i) = q_x(i) + q_y(i) - [q_x(i)q_y(i)]/cpw_{xy}(i), \tag{3.4}$$

where $cpw_{xy}(i)$ is the product of common path weights from $x$ and $y$ to $i$.

Consequently, we use depth first search to find a common path in aggregating any two vectors, as shown in Alg. 3. The algorithm finds a common path from the seed set $S$ and candidate $c$ to *targetNode*, where *targetNode* is the starting node for the search. $q_c$ is the SSS vector for the candidate $c$, and $q_S$ is a vector representing all seed vectors aggregated. We note that $q_S$ can represent one or more seeds, where $S$ in $q_S$ denotes the set of seeds. *hopLimit* is the maximum hop parameter. The output is the product of the weights on a common path from the seeds to the target node, or 1.0 if a common path does not exist. The algorithm starts by getting the in-neighbors ($N_{in}$) of the target node (e.g. node $z$ in Fig. 3.1). It does not consider a neighbor that is not a descendant of both $c$ and nodes in $S$ (line 7). Otherwise, if the neighbor is a descendant of both, *currentNode* is updated by the node ID of the neighbor (line 9). We use two stopping criteria for the while loop (line 4). First, the number of hops is limited to *hopLimit*. This is useful because we restrict the number of hops when calculating the SSS vectors (discussed in section 3.4). Second, the loop stops when it reaches to the candidate node $c$. This is because the effect of previous seeds on the common path is already calculated. Therefore, we follow a common path until we reach to the candidate node. The *break* statement (line 10) limits the search to finding only a single path. Because as discussed before, we find only one common path weight to improve the accuracy of the estimation.

As we discussed in Section 3.3.2, when *targetNode* (node $i$) is not influenced by either or both $c$ (node $x$) and $S$ (node $y$), we do not need Eq.3.2. Therefore, Alg. 3 is not called to adjust Eq.3.2. If *targetNode* is influenced by both $c$ and $S$, adjustment to Eq.3.2 might be needed and Alg. 3 is called. When a common path is not found, the algorithm returns 1.0 and adjustment is not applied to the estimation.

#### 3.3.2.2 Ancestor Checking for Blocked Seeds

Before we aggregate SSS-vectors, we need to check whether any seed is *blocked* by another seed and update the SSS-vectors if necessary. PMIA [27] and IPA [78] for the IC model also consider blocked seeds. Node $v$ is an ancestor of node $u$ if there exists a path from $v$ to $u$ in the graph. Similarly, $u$ is considered a descendant of $v$. Since if a path from node $v$ to node $u$ exists, the spread from $v$ to $u$ is larger

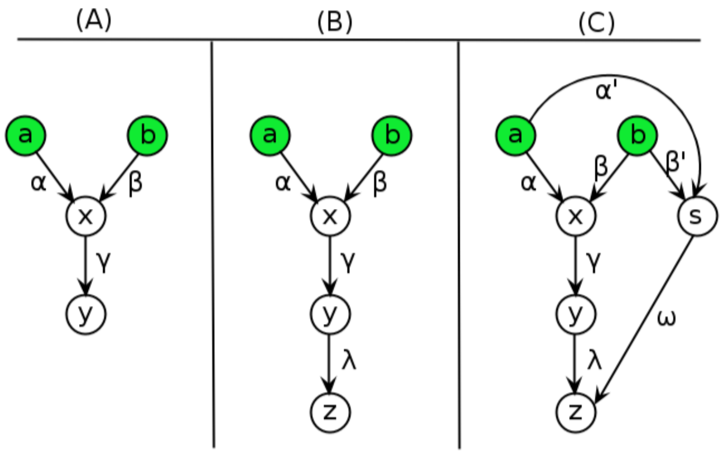


Figure 3.1: 3 Examples showing common paths in which 2 seeds flow (Seeds are green, other nodes are white).

**Algorithm 3** CommonPathWeights($targetNode$, $c$, $q_c$, $q_S$, $hopLimit$)

```
1:  $hopCount \leftarrow 0$
2:  $cpw \leftarrow 1.0$
3:  $currentNode \leftarrow targetNode$
4:  while $hopCount < hopLimit$ and $c \notin N_{in}(currentNode)$ do
5:      $hopCount \leftarrow hopCount + 1$
6:      for $u, w \in N_{in}(currentNode)$ do        ▷ w is the weight from an in-neighbor to currentNode
7:          if $q_c(u) > 0$ and $q_S(u) > 0$ then ▷ $u$ is a descendant of $c$ and the seeds in $S$
8:              $cpw \leftarrow cpw * w$
9:              $currentNode \leftarrow u$
10:             $break$
    return $cpw$
```

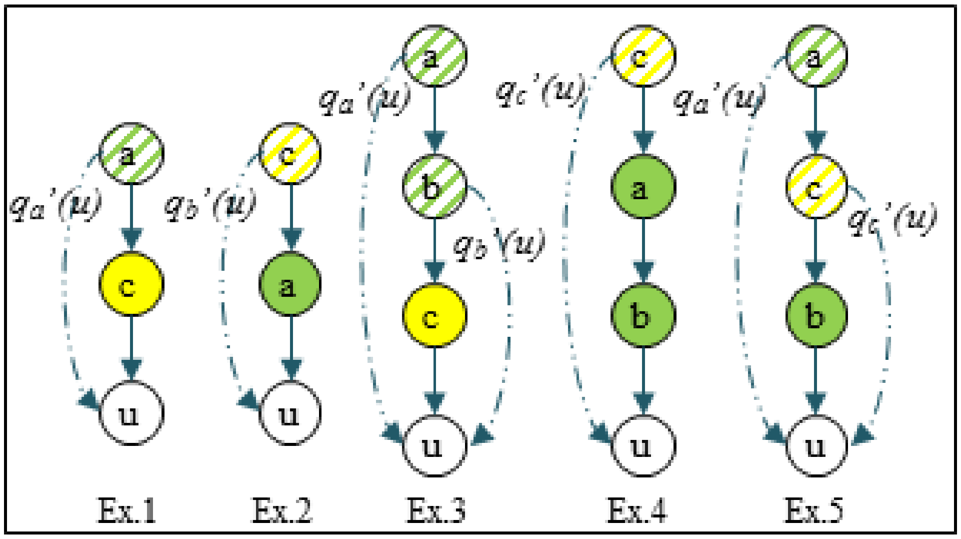


Figure 3.2: Examples of Blocked seeds (seeds in green, candidates in yellow, blocked nodes in stripes, modified spread of blocked nodes in dashed paths)

than zero. Hence, to check if $v$ is an ancestor of $u$, we check if $q_v(u)$ is positive. Given a seed $s$, a candidate seed $c$, and a non-seed node $v$, $s$ is considered blocked by $c$ with respect to $v$ if $s$ is an ancestor of $c$, and there exists a path from $s$ to $v$ that passes through $c$. Similarly, $c$ is considered blocked by $s$ with respect to $v$ if $c$ is an ancestor of $s$, and there exists a path from $c$ to $v$ that passes through $s$.

When a seed $s$ is blocked with respect to a node $v$, the spread to $v$ from $s$ is reduced. Not considering the blocked nodes can lead to eSSS overestimating the actual SSS. We discuss how to update the SSS-vectors of blocked nodes with five examples depicted in Fig. 3.2.

Example 1: Consider $S$ contains a single seed $a$, and candidate $c$ is being added. Here we aim to calculate $eSSS(\{a, c\})$. Suppose that $a$ is blocked by $c$ (w.r.t $u$). Since $c$ becomes a seed ($c$ contains the information), $a$ cannot influence $c$ and $a$ cannot influence $u$ via $c$ any more. However, $a$ can influence $u$ via other paths not containing $c$, which is a discounted spread from $a$ to $c$. That is, we need to update the spread to $u$ in the SSS-Vector of $a$ to the discounted spread.

Before $c$ becomes a seed, let $\omega$ be the spread from $a$ to $u$, which is $q_a(u)$. $\omega$ is the aggregate of two components: spread from $a$ via $c$ and spread from $a$ *not* via $c$. Let $\alpha$ be the spread from $a$ to $c$, which is $q_a(c)$, and $\beta$ be the spread from $c$ to $u$, which is $q_c(u)$. We estimate the first component by $\alpha\beta$. Let $\gamma$ be the second component, which is the discounted spread when $c$ becomes a seed. Figure 3.2 depicts this example. Since $\omega$ is the aggregate spread of the two components, from Eq. 3.2:

$$\begin{aligned} \omega &= 1 - (1 - \alpha\beta) \times (1 - \gamma) = \alpha\beta + \gamma - \alpha\beta\gamma \\ \gamma &= \frac{\omega - \alpha\beta}{1 - \alpha\beta} \end{aligned} \tag{3.5}$$

That is, we update $q_a(u)$ to the discounted spread $\gamma$ when $c$ becomes a seed. Since the transmission probabilities are usually less than 1, $\alpha$ and $\beta$ are generally less than 1 and the denominator in Eq. 3.5 generally cannot be zero. If $\alpha$ and $\beta$ are both 1, the value of $q_a(u)$ is not important since $q_c(u)$ (spread from $c$ to $u$) is 1 and the total spread from all nodes to $u$ cannot exceed 1.

To generalize the calculation, $\omega$ represents the spread from a blocked seed to a node $u$ before considering candidate $c$. $\alpha$ denotes the spread from the blocked seed to the blocking seed and $\beta$ the spread from the blocking seed to $v$. $\gamma$ is the spread from the blocked seed to $u$ via paths not involving the blocking seed and is hence the updated (discounted) spread from the blocked seed to $u$ after considering candidate $c$.

Example 2: Seed $a$ is an ancestor of candidate $c$ in Example 1, we now consider the opposite case when $c$ is an ancestor of $a$. That is, $c$ is blocked by $a$ and the SSS-vector of $c$ need to be updated. The blocked node is $c$ and the blocking node is $a$. Hence, $\omega$ is $q_c(u)$, $\alpha$ is $q_c(a)$, $\beta$ is $q_a(u)$, and $q_c(u)$ is updated to be $\gamma$ in

Eq. 3.5.

Example 3: A more complicated case is when there are multiple seeds in $S$, and $c$ can be ancestor/descendant of multiple of them. Consider set $S$ containing nodes $a, b$ and candidate node $c$. Also, $a$ was added to $S$ before $b$. Here we aim to calculate $eSSS(\{a, b, c\})$. Suppose that $a$ is an ancestor of $b$ and $b$ is an ancestor of $c$. That is, $a$ is blocked by $b$, and $b$ is blocked by $c$ (w.r.t $u$). Since $a$ is blocked by $b$, the spread from $a$ to $u$ has been updated to the spread via paths not involving $b$ when $b$ was a candidate previously. When $c$ becomes a candidate, we need to further update the spread of $a$ to $u$ via paths not involving $c$ — $\omega$ is $q_a(u)$, $\alpha$ is $q_a(c)$, $\beta$ is $q_c(u)$, and $q_a(u)$ is updated to be $\gamma$ in Eq. 3.5. Similarly, we need to update the spread of $b$ to $u$ via paths not involving $c$ — $\omega$ is $q_b(u)$, $\alpha$ is $q_b(c)$, $\beta$ is $q_c(u)$, and $q_b(u)$ is updated to be $\gamma$ in Eq. 3.5.

Example 4: Consider $a$ is an ancestor of $b$ as in Example 3, but now $c$ is an ancestor of $a$. That is, $c$ is blocked by $a$ and $a$ is blocked by $b$. Since $a$ is blocked by $b$, the spread from $a$ to $u$ has been updated to the spread via paths not involving $b$ when $b$ was a candidate previously. When $c$ becomes a candidate, we need to update the spread of $c$ to $u$ via paths not involving $a$ — $\omega$ is $q_c(u)$, $\alpha$ is $q_c(a)$, $\beta$ is $q_a(u)$, and $q_c(u)$ is updated to be $\gamma$ in Eq. 3.5. Though $a$ is an ancestor of $b$, $c$ might have paths to $u$ via $b$ but not $a$. Hence, we also need to update the spread of $c$ to $u$ via paths not involving $b$ — $\omega$ is $q_c(u)$, $\alpha$ is $q_c(b)$, $\beta$ is $q_b(u)$, and $q_c(u)$ is updated to be $\gamma$ in Eq. 3.5.

Example 5: Consider $a$ is an ancestor of $b$ as in Example 3, but now $c$ is "between" $a$ and $b$. That is, $a$ is blocked by $c$ and $c$ is blocked by $b$. Similar to Examples 3 and 4, the spread from $a$ to $v$ has been updated to the spread via paths not involving $b$ when $b$ was a candidate previously. When $c$ becomes a candidate, there might be paths from $a$ to $u$ involving $c$, so we need to update the spread of $a$ to $u$ via paths not involving $c$ — $\omega$ is $q_a(u)$, $\alpha$ is $q_a(c)$, $\beta$ is $q_c(u)$, and $q_a(u)$ is updated to be $\gamma$ in Eq. 3.5. Also, we need to update the spread of $c$ to $u$ via paths not involving $b$ — $\omega$ is $q_c(u)$, $\alpha$ is $q_c(b)$, $\beta$ is $q_b(u)$, and $q_c(u)$ is updated to be $\gamma$ in Eq. 3.5. We update $q_a(u)$ and $q_c(u)$, but $q_c(u)$ is used to update $q_a(u)$. Based on some experiments, we choose to update $q_a(u)$ using the original $q_c(u)$, not the updated $q_c(u)$, to reduce error.

The above five examples help illustrate the general case, where we check if candidate $c$ is an ancestor or descendant of each seed in $S$. If so, we use Eq. 3.5 to update the SSS-Vector of each blocked seed. This process helps increase the accuracy of eSSS. Alg. 4 illustrates how SSS-vectors are modified when blocking exists between the seeds and candidate. Parameter $c$ is the candidate, $S$ is the seed set, $q_S$ is the aggregated vector for $S$, and $q$ has the vector of each node in the graph. If the candidate is blocked by a seed, we update the vector for the candidate according to Eq. 3.5 (lines 4-8). If a seed is an ancestor of the candidate, we update the vector for the seed similarly (lines 9-13). Note that a cycle could exist that involves $c$ and $v$, so we check for ancestors both ways in lines 4 and 9. When a seed or a candidate is not blocked, their vectors need not be modified (lines 15 and 17). Vectors for the seeds and candidate are returned.

In summary, our approach to estimating SSS is to first find blocked seeds and then update the SSS-Vectors of the blocked seeds (Eq. 3.5). Next, the vectors are aggregated (Eq. 3.3), with adjustments from common path weights (Eq. 3.4). The resulting SSS-Vector ($q_S$) is summed over all elements to obtain the eSSS value: $eSSS(S) = \sum_i^n q_S(i)$.

We note that *eSSS* might not estimate SSS perfectly. We check ancestors and

**Algorithm 4** BlockedVec($c, S, q_S, q$)

```
1: $c.blocked \leftarrow false$
2: for $v \in S$ do
3:     $v.blocked \leftarrow false$
4:     if $c$ is an ancestor of $v$ then                    $\triangleright$ $c$ is blocked by $v$
5:         $c.blocked \leftarrow true$
6:         for $u \in \{i|q_c(i) > 0\}$ do
7:             $\alpha\beta \leftarrow q_c(v) \cdot q_v(u)$
8:             $q'_c(u) \leftarrow (q_c(u) - \alpha\beta)/(1 - \alpha\beta)$
9:     if $v$ is an ancestor of $c$ then                    $\triangleright$ $v$ is blocked by $c$
10:        $v.blocked \leftarrow true$
11:        for $u \in \{i|q_v(i) > 0\}$ do
12:            $\alpha\beta \leftarrow q_v(c) \cdot q_c(u)$
13:            $q'_v(u) \leftarrow (q_v(u) - \alpha\beta)/(1 - \alpha\beta)$
14:    if $v.blocked = false$ then
15:        $q'_v \leftarrow q_v$
16: if $c.blocked = false$ then
17:     $q'_c \leftarrow q_c$
18: return $[q'_{v_1}, ...q'_{v_{|S|}}, q'_c]$
```

update vectors only once before aggregating two vectors. However, in case of cycles involving seeds, the vectors should be updated multiple times until convergence. Though, the likelihood of cycles is small within 3 hops. Also, we find the weights of only one common path. However, multiple common paths might exist. Although additional computation can yield a more accurate estimation, we aim at a more efficient and relatively accurate estimation.

### 3.3.3 Improving efficiency of the Greedy Algorithm

The greedy algorithm calculates $eSSS(S \cup \{c\})$ for each candidate $c$ and the gain efficiently: $gain(c, S) = eSSS(S \cup \{c\}) - eSSS(S)$. Because of the submodularity property of eSSS, we can improve the efficiency by not updating the gain of every candidate [88].

**Theorem 1.** *eSSS is submodular. The submodularity property states:*

*gain* $(c, S\cup\{x\}) \leq gain(c,S)$, *where $x$ is an additional seed (the newly added seed in our case) and $c$ is a candidate. That is, the gain for candidate $c$ cannot be larger than the gain obtained from the previous level(s). (We consider the greedy algorithm conducts a tree-like search and selects one seed at each level.)*

*Proof.* eSSS is submodular due to two reasons. First, the additional seed in $S$ can block other seeds, and hence can reduce the spread in their SSS-vectors. Second, from Eq.3.3, the gain of adding $c$ to $S$ at node $i$ is:

$$\begin{aligned} gain(c,S,i) &= \Big(1-(1-q_c(i))\prod\nolimits_j^k \big(1-q_{v_j}(i)\big)\Big) \\ &\quad - \Big(1-\prod\nolimits_j^k (1-q_{v_j}(i))\Big) \\ &= q_c(i)\prod\nolimits_j^k (1-q_{v_j}(i)). \end{aligned}$$

Similarly, the gain of adding $c$ to $S\cup\{x\}$ at node $i$ is:

$$gain(c, S\cup\{x\}, i) = q_c(i)(1-q_x(i))\prod\nolimits_j^k (1-q_{v_j}(i))$$

Since $0 \leq (1-q_x(i)) \leq 1$:

$$gain(c, S\cup\{x\}, i) \leq gain(c, S, i).$$

After summing the gain at each node $i$:

$$gain(c, S\cup\{x\}) \leq gain(c, S).$$

That is, aggregating SSS-vectors has diminishing gain for the same candidate $c$ as $S$ grows. □

To utilize the submodularity property of eSSS for reducing computation, we use a priority queue to store the gain of each candidate and the level number when the gain was updated [52]. If the largest *updated* gain at the current level is larger than the largest *non-updated* gain, we prune the gain updates for the rest of the candidates, which cannot yield a larger updated gain. Hence, if the candidate at the head of the priority queue has been updated at the current level, we select it and add it to the seed set.

### 3.3.4 VSM Algorithm

Our VSM (Vector-based Spread Maximization) algorithm uses the Greedy algorithm (Alg. 2) with eSSS as the evaluation function $f$. Since eSSS is non-negative, monotone, and submodular (Sec. 3.3.3), VSM guarantees that the found solution is at least $1 - 1/e$ (63%) of the optimal solution based on eSSS (Theorem 2.1 in [73]). Though eSSS is an estimate of SSS, which is the objective function, our empirical results indicate that eSSS is within 0.12% of SSS (Sec. 3.5.8).

Alg. 5 illustrates our VSM algorithm. VSM finds the SSS values for each vertex using Alg. 6 (an improved version of SSS, which is discussed in Sec. 3.4), saves the SSS-vectors, and populates the priority queue (lines 1-4). We initialize the seed set, aggregated SSS-vector of the seed set, and eSSS value of the seed set (lines 5-8). While the candidate's level is less than the current level, its gain is not up to date (line 13). If the candidate is an ancestor or descendant of any seed in the seed set, the SSS-vectors are modified and the aggregated vector is updated (line 14-17). Otherwise, vectors of the seeds and candidate are not modified, we update the aggregated vector by aggregating existing aggregated vector of the seed set and vector of the candidate (line 19). We update the gain and level of the candidate in the priority queue and heapify the priority queue (lines 10-22). If the candidate is still at the head of the priority queue, the updated candidate is the best candidate and we do not use the previously saved vectors (lines 23-24). Otherwise, if the candidate's gain is larger than the largest gain so far, we save the modified vectors so that we do not need to recalculate them if the candidate eventually becomes the best candidate (lines 25-28). We remove the best candidate from the priority queue, update the vectors from the modified versions if needed, update the eSSS value of the seed sets, and add the best candidate to the seed set (lines 29-37).

#### 3.3.4.1 Improving space and time

Alg. 5 generates a vector for each node (lines 1 - 4). For a graph with $n$ nodes, the space complexity is $O(n^2)$, which could be prohibitive for large graphs. Based on initial experiments, we observe that many of the vectors are not used because of pruning in Sec. 3.3.3. Generally, fewer than $2k$ ($k$ is the seed set size) vectors are used. Hence, an improved VSM generates vectors for only $2k$ nodes initially to reduce space and time. To select the initial $2k$ nodes, we find the nodes with the highest spread to their immediate out-neighbors. If needed (when the priority queue is exhausted), vectors for additional nodes are generated in the order of spread to their immediate neighbors. IPA [78] for the IC model similarly reduces space by limiting the priority queue to be of size $3k$. However, their approach does not allow further expansion of the priority queue if additional candidate nodes can improve gain. Moreover, IPA stores paths from each node, which requires more space than spread values from each node in VSM.

To calculate $q_{S'}$ for the updated $S$, line 17 of Alg. 5 aggregates all the seed vectors. However, not all seed vectors are updated since some seeds are not blocked

**Algorithm 5** VSM($V, P, k$)

```
1: for $v \in V$ do
2:     $[q_v, v.gain] \leftarrow SSS2(\{v\}, P)$                              ▷ Alg. 6
3:     $v.level \leftarrow 1$
4:     $PQ.insert(v)$                                         ▷ priority queue wrt gain
5: $v \leftarrow PQ.remove()$
6: $S \leftarrow \{v\}$                                                    ▷ seed set
7: $q_S \leftarrow q_v$                                    ▷ aggregated vector of seed set
8: $eSSSofS \leftarrow v.gain$                                 ▷ eSSS value of seed set
9: for $level \leftarrow 2$ to $k$ do                                     ▷ $k$ seeds
10:     $bestGain \leftarrow 0$
11:     $bestVec \leftarrow \emptyset$
12:     $c \leftarrow PQ.head()$                                          ▷ candidate $c$
13:     while $c.level < level$ do                      ▷ $c$'s gain is not up to date
14:         $c.blockedSeeds \leftarrow false$
15:         if $c$ is ancestor/descendant of $v, v \in S$ then
16:             $c.blockedSeeds \leftarrow true$
17:             $[q'_{v_1} ... q'_{v_{|S|}}, q'_c] \leftarrow BlockedVec(c, S, q_S, q)$
18:             $q_{S'} \leftarrow q'_{v_1} \oplus ... \oplus q'_{v_{|S|}} \oplus q'_c$
19:         else
20:             $q_{S'} \leftarrow q_S \oplus q_c$
21:         $c.gain \leftarrow eSSS(q_{S'}) - eSSSofS$
22:         $c.level \leftarrow level$
23:         $PQ.heapify()$
24:         if $PQ.head() = c$ then                        ▷ updated gain of $c$ is best
25:             $bestVec \leftarrow \emptyset$
26:         else if $c.gain > bestGain$ then
27:             $bestGain \leftarrow c.gain$
28:             $bestVec \leftarrow [q'_{v_1}, ..., q'_{v_{|S|}}, q'_c, q_{S'}]$
29:         $c \leftarrow PQ.head()$
30:     $c \leftarrow PQ.remove()$                             ▷ best candidate is found
31:     if $bestVec \neq \emptyset$ then
32:         $[q_{v_1}, ..., q_{v_{|S|}}, q_c, q_S] \leftarrow bestVec$
33:     else if $c.blockedSeeds = true$ then
34:         $[q_{v_1}, ..., q_{v_{|S|}}, q_c, q_S] \leftarrow [q'_{v_1}, ..., q'_{v_{|S|}}, q'_c, q_{S'}]$
35:     else
36:         $q_S \leftarrow q_{S'}$
37:     $eSSSofS \leftarrow eSSSofS + c.gain$
38:     $S \leftarrow S \cup c$
39: return $S$
```

by another seed. If the number of updated seed vectors is small, we can reduce computation. We use $\ominus$ to denote the deaggregate operator and follow Eq 3.2:

$$\begin{aligned} q_x &= q_{\{x,y\}} \ominus q_y \\ q_x(i) &= (q_{\{x,y\}}(i) - q_y(i))/(1 - q_y(i)). \end{aligned} \tag{3.6}$$

Since the transmission probabilities are usually less than 1, $q_y(i)$ is generally less than 1 and the denominator in Eq. 3.6 generally cannot be zero. If $q_y(i)$ is 1, the value of $q_x(i)$ is not important since $q_y(i)$ (spread from $y$ to $i$) is 1 and the total spread from all nodes to $i$ cannot exceed 1.

Consider only seed $v$ is blocked by candidate $c$, $q_v$ is the vector before adding $c$, and $q'_v$ is the updated vector afterwards. We can calculate $q_{S'}$ as: $q_S \ominus q_v \oplus q'_v \oplus q_c$, instead of aggregating all the seed vectors from scratch. Let $b$ be the number of blocked seeds. Aggregating the seed vectors from scratch needs $|S|-1$ aggregations. Deaggregating and aggregating the updated seed vectors needs $b$ deaggregations and $b$ aggregations. If $|S|-1 < 2b$, VSM aggregates vectors from scratch; otherwise, it de/aggregates updated vectors.

The nested loop starting on line 9 dominates VSM's time—outer loop runs $O(k)$ times and inner loop runs $O(n)$ times [but $O(k)$ in practice due to pruning (Sec.3.3.3)]. At each iteration of the inner loop, $O(kn)$ for $BlockedVec()$, $O(kn)$ for aggregating vectors, $O(\log n)$ for $heapify()$, and $O(kn)$ for copying into $savedVec$. Hence, VSM's time complexity is $O(k \cdot n \cdot kn)$ or $O(k^2n^2)$ [but $O(k^3n)$ in practice]. Since VSM stores vectors for $2k$ nodes (the dominant data structure), the space complexity is $O(kn)$.

## 3.4 More efficient SSS and GreedySSS

The SSS method in Alg. 1 updates the $q$ value for all nodes except the seeds, however, many of them will remain zero, particularly in a large sparse graph. To improve the efficiency of SSS for large sparse graphs, we only update nodes that will have positive spread. We call these nodes "activated" nodes. We use the outgoing edges of the activated nodes of the previous iteration to find the activated nodes of the current iteration. Also, we keep track of newly activated nodes in the previous iteration so that we only need to add their out-neighbors as activated nodes in the current iteration, otherwise we unnecessarily add out-neighbors that have been added in the previous iterations. We only consider activated nodes for updating $q$, total spread, and change in total spread. Moreover, the number of activated nodes can grow exponentially. To prevent finding a large number of activated nodes and not using them in the last iteration, we find activated nodes at the beginning of the loop for the current iteration rather than at the bottom of the loop for the next iteration.

Alg. 6 shows the improved algorithm called SSS2. We initialize the sets for activated nodes, newly activated nodes and old ones (lines 4-6). The activated nodes are updated to be the union of the old ones and out-neighbors of the newly activated nodes from the previous iteration. We then exclude the seeds and find the newly activated nodes (lines 10-12). We update $q$ and $C$, and return the total spread considering only the activated nodes (lines 13, 14 and 17). For further efficiency, we initialize $q$ on line 2 based on the activated nodes in the previous SSS2 call (not included in Alg. 6).

From seed $s$ to a node $i$, the more hops there are between $s$ and $i$, the smaller the spread is from $s$ to $i$ because of the discount from propagation probability in

**Algorithm 6** SSS2($S$, $P, hoplimit$)

```
1: $\forall i \in S \quad q^0(i) \leftarrow 1$
2: $\forall i \notin S \quad q^0(i) \leftarrow 0$
3: $t \leftarrow 0$
4: $A \leftarrow outNeighbors(S)$                    ▷ activated nodes
5: $A_{new} \leftarrow A$                             ▷ newly activated nodes
6: $A_{old} \leftarrow \emptyset$                     ▷ activated in previous iteration
7: repeat
8:     $\forall i \in S, \quad q^{t+1}(i) \leftarrow 1$
9:     if $t > 0$ then
10:        $A_{old} \leftarrow A$
11:        $A \leftarrow (A_{old} \cup outNeighbors(A_{new}))\backslash S$
12:        $A_{new} \leftarrow A \backslash A_{old}$
13:    $\forall i \in A, \quad q^{t+1}(i) \leftarrow 1 - \prod_{j \in N(i)}(1 - P_{ji} \cdot q^t(l))$
14:    $C_{t+1} \leftarrow \sum_{i \in A} |q^{t+1}(i) - q^t(i)|$
15:    $t \leftarrow t + 1$
16: until $C_t < 0.01 \cdot C_1$ or $t \geq hoplimit$
17: return $\sum_{i \in A} q^t(i)$
```

each hop. Goyal et al. [53] observe that much of the spread is within 3 or 4 hops from the seeds. For efficiency, we stop updating $q$ if the hop limit is exceeded (line 19). Note that when there is a hop limit, the return value might not closely estimate the steady state value since the convergence criterion of less than 1% change might not have met. Hence, when SSS is used as the objective function for evaluating and comparing different algorithms, we do *not* use a hop limit.

Since SSS2 is faster than SSS, we propose GreedySSS, which is the same as VSM, except that it calls SSS2 (Alg. 6) instead of estimating SSS from SSS-vectors. We would like to see if GreedySSS is more effective (but slower) than VSM because SSS values are not estimated.

## 3.5 Experimental Evaluation

### 3.5.1 Experimental Criteria

The main evaluation criterion is effectiveness as measured by SSS2 (Alg. 6) *without a hop limit*. To evaluate efficiency, we measure the CPU running time. To evaluate the accuracy of eSSS, we measure the % difference, which is $(eSSS - SSS)/SSS * 100\%$. To evaluate the amount of storage for the SSS-vectors, we measure the number of (positive) entries in the SSS-vectors.

### 3.5.2 Experimental Data and Procedures

We use three datasets: DBLP, Last.fm, and Twitter from Aggarwal el. al [4]. DBLP has 684,911 authors and 7,764,604 edges. Last.fm has 818,800 users and 3,340,954 friendships. Twitter has 1,994,092 users and 6,450,193 edges.

We evaluate our proposed VSM (Sec. 3.3) and GreedySSS (Sec. 3.4), and compare them with RankedReplace [4] and Bayes Traceback [4]. For VSM, we evaluate two versions: with or without ancestor checking for blocked seeds. We varied $k$ from 20 to 100, with an increment of 20 as in [4]. VSM, GreedySSS and RankedReplace need to calculate SSS and we use our faster SSS2 algorithm (Alg. 6) for a comparison that focus on differences not contributed by the improvement due to SSS2. The hop limit for SSS2 is 3. The replacement factor $r$ is 10 for RankedReplace. The discard fraction $f$ for Bayes Traceback is 0.25, 0.2, and 0.3 for DBLP, Last.fm and Twitter respectively (the parameters were selected to maximize effectiveness). The algorithms were implemented in Python. The implementations were run on a 128GB, 8-core virtual machine running on ESXi 6.0.0 on a Dell PowerEdge M620 containing 2x Intel Xeon E5-2630 V2 @ 2.6 GHz

Table 3.1: Running time of SSS2 vs. SSS (seconds) on 100,000 nodes

| Alg. | DBLP | LAST.FM | Twitter |
|---|---|---|---|
| SSS (Alg. 1) | 16524.74 | 18512.06 | 12006.22 |
| SSS2 (Alg. 6) | 0.87 | 1.38 | 0.57 |

with Ubuntu Linux 14.04.

### 3.5.3 Efficiency of SSS2

To compare the efficiency of our improved SSS2 (Alg. 6) with SSS (Alg. 1), we sampled 100,000 nodes from the three datasets and measured the running time of the two algorithms calculating SSS (without a hop limit) for all vertices in the subsets. The results in Table 3.1 indicate that our proposed improvement is about 4 orders of magnitude faster.

### 3.5.4 Selecting Hop Limit for SSS2 with Smaller Datasets

Our experiments with smaller datasets of 30K nodes indicate that VSM with hop limits of 2 and 3 achieves the same SSS as VSM with no hop limits (Table 3.2). Interestingly, with and without ancestor checking for blocked seeds also yield the same SSS when the hop limit is 2, 3, or none. In terms of running time (not shown due to space limitation), raising the hop limit increases computation. As $k$ increases, computation grows faster with ancestor checking than without ancestor checking. Interestingly, the increase in computation from a hop limit of 3 to none is much smaller than the increase from a hop limit of 2 to 3. For Last.fm and Twitter, the increase in computation from a hop limit of 3 to none is quite small. This indicates that a hop limit of 3 is close to convergence, which is used as a stopping criterion when hop limit is none. In summary, a hop limit of 2 or 3, with

Table 3.2: Hop limit vs. SSS values (30k nodes) (AC: Ancestor Checking)

| Hop | AC | $k$=20 | $k$=40 | $k$=60 | $k$=80 | $k$=100 |
|---|---|---|---|---|---|---|
| DBLP | | | | | | |
| 1 | false | 36.51 | 66.94 | 95.52 | 122.8 | 149.2 |
| 2 | false | 36.54 | 67.04 | 95.47 | 122.8 | 149.2 |
| 3 | false | 36.54 | 67.04 | 95.46 | 122.8 | 149.1 |
| no limit | false | 36.54 | 67.04 | 95.46 | 122.8 | 149.1 |
| 1 | true | 36.51 | 66.81 | 95.52 | 122.7 | 149.2 |
| 2 | true | 36.56 | 67.04 | 95.58 | 122.8 | 149.2 |
| 3 | true | 36.56 | 67.04 | 95.57 | 122.8 | 149.2 |
| no limit | true | 36.56 | 67.04 | 95.57 | 122.8 | 149.2 |
| LAST.FM | | | | | | |
| 1 | false | 61.8 | 101.7 | 136.9 | 170.6 | 203.5 |
| 2 | false | 61.8 | 101.8 | 137.2 | 171.0 | 203.7 |
| 3 | false | 61.8 | 101.8 | 137.2 | 171.0 | 203.7 |
| no limit | false | 61.8 | 101.8 | 137.2 | 171.0 | 203.7 |
| 1 | true | 61.8 | 101.7 | 136.9 | 170.6 | 203.5 |
| 2 | true | 61.8 | 101.8 | 137.2 | 171.0 | 203.8 |
| 3 | true | 61.8 | 101.8 | 137.2 | 171.0 | 203.8 |
| no limit | true | 61.8 | 101.8 | 137.2 | 171.0 | 203.8 |
| Twitter | | | | | | |
| 1 | false | 33.86 | 62.10 | 89.54 | 116.1 | 141.9 |
| 2 | false | 33.87 | 62.23 | 89.57 | 116.1 | 142.0 |
| 3 | false | 33.87 | 62.23 | 89.57 | 116.1 | 142.0 |
| no limit | false | 33.87 | 62.23 | 89.57 | 116.1 | 142.0 |
| 1 | true | 33.86 | 62.10 | 89.54 | 116.1 | 141.9 |
| 2 | true | 33.87 | 62.23 | 89.57 | 116.2 | 142.0 |
| 3 | true | 33.87 | 62.23 | 89.57 | 116.2 | 142.0 |
| no limit | true | 33.87 | 62.23 | 89.57 | 116.2 | 142.0 |

Table 3.3: SSS of algorithms bold: highest, underline: $\leq$ 0.1% from highest)

| Algorithm | $k$=20 | $k$=40 | $k$=60 | $k$=80 | $k$=100 |
|---|---|---|---|---|---|
| DBLP | | | | | |
| RankedReplace | 97.8 | 170.9 | 236.6 | 297.8 | 355.3 |
| BayesTraceback | 67.8 | 123.7 | 170.7 | 224.9 | 298.8 |
| GreedySSS | 97.8 | 171.0 | 236.6 | 297.9 | 355.6 |
| VSM ac=false | 97.7 | 170.2 | 236.4 | 298.2 | <u>355.9</u> |
| VSM ac=true | **98.2** | **171.5** | **237.3** | **298.7** | **356.0** |
| LAST.FM | | | | | |
| RankedReplace | 568.4 | 816.8 | 1024.2 | 1210.6 | 1377.9 |
| BayesTraceback | 332.6 | 545.8 | 708.1 | 816.2 | 952.1 |
| GreedySSS | 568.4 | 816.8 | 1024.2 | 1210.6 | 1377.9 |
| VSM ac=false | **571.6** | **822.4** | **1033.0** | **1221.5** | <u>1389.7</u> |
| VSM ac=true | **571.6** | **822.4** | 1031.6 | **1221.5** | **1390.1** |
| Twitter | | | | | |
| RankedReplace | 314.9 | 489.0 | 625.0 | 742.9 | 847.6 |
| BayesTraceback | 189.7 | 311.0 | 414.6 | 522.5 | 595.9 |
| GreedySSS | 314.9 | 489.0 | 625.1 | 742.9 | 847.6 |
| VSM ac=false | <u>315.2</u> | **489.7** | <u>625.9</u> | <u>743.8</u> | 848.7 |
| VSM ac=true | **315.3** | **489.7** | **626.0** | **744.0** | **849.7** |

less computation, yields the same effectiveness as no hop limit for datasets with 30K nodes. We conservatively choose 3 as the default hop limit.

### 3.5.5 Effectiveness of Algorithms

Table 3.3 displays the effectiveness of different algorithms. Generally, VSM with ancestor checking for blocked seeds is more effective than without ancestor checking. For DBLP, VSM with ancestor checking outperforms the other algorithms consistently. For Twitter, VSM with ancestor checking outperforms the other algorithms. For Last.fm, VSM with ancestor checking outperforms the others, except when $k$=60. Interestingly, GreedySSS is generally less effective than VSM with ancestor checking, even though VSM estimates SSS, and GreedySSS measures SSS. One reason might be SSS with a low hop limit has not converged, while eSSS is more accurate in adjusting SSS-vectors for blocked seeds. Overall, VSM with ancestor checking is more effective than the other algorithms across the three datasets.

Some of the SSS values are similar to the highest value—at most 0.1% difference from the highest value. For DBLP, when $k$=100 VSM without ancestor checking is similar to the most effective algorithm. For the Twitter dataset, VSM without ancestor checking has similar SSS values as the most effective algorithm at $k$= 20, 60, 80. For Last.fm, the two versions of VSM are similar to the most effective algorithm, except for VSM with ancestor checking at $k$=60. Overall, compared to VSM, BayesTraceback is significantly less effective, while the other algorithms are within 1% difference in effectiveness across the three datasets.

### 3.5.6 Efficiency of Algorithms

Figure 3.3, 3.4, and 3.5 plot the running time of different algorithms. VSM with ancestor checking (hop=3) is about an order of magnitude faster than RankedReplace and VSM without ancestor checking (hop=3) is about 2 orders of magnitude faster. VSM without ancestor checking (hop=3) is generally faster (and more effective) than Bayes Traceback. Since GreedySSS measures SSS instead of estimating SSS, it is generally slower than VSM as expected. Note that we use our proposed SSS2 algorithm (Alg. 6) in RankedReplace in all our experiments. If we use the original SSS (Alg. 1), RankedReplace will be much slower (Sec. 3.5.3).

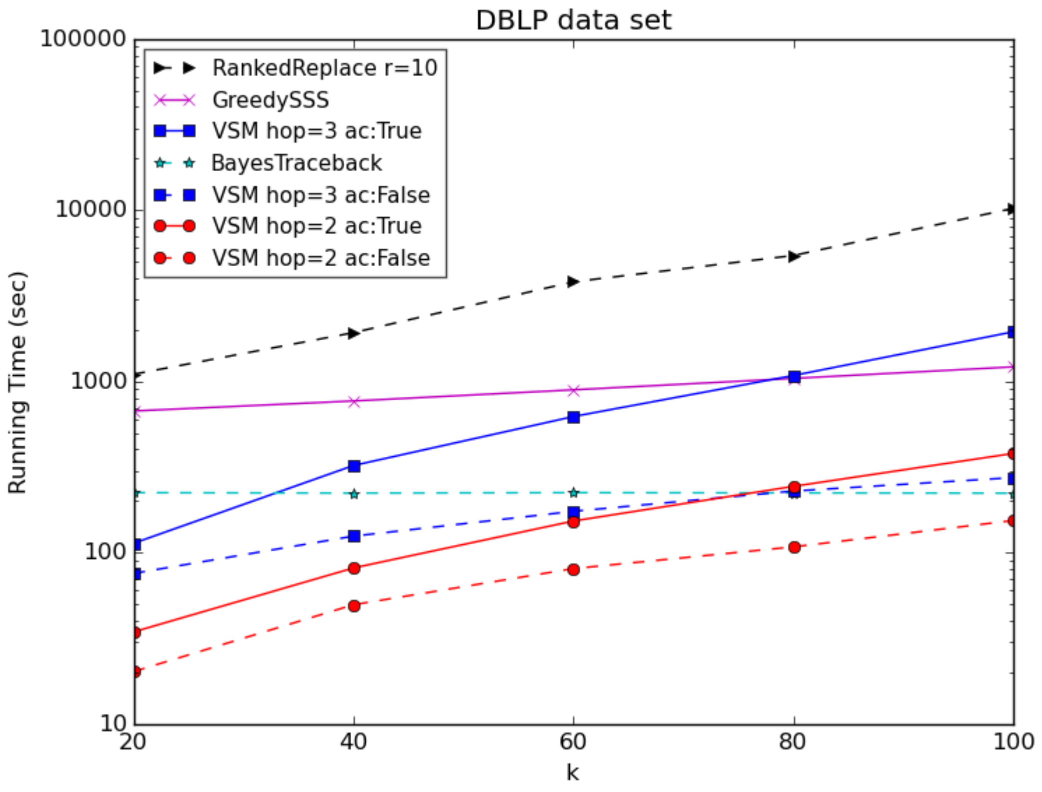


Figure 3.3: Efficiency versus $k$ on DBLP

### 3.5.7 Effectiveness, Efficiency, and Space in VSM

The effectiveness of VSM with hop limits from 2 to 3 is displayed in Table 3.4. Generally, increasing the hop limit increases the effectiveness. The difference between hop limits of 2 and 3 is at most 0.2% and sometimes non-existent, which is similar to our earlier experiments with smaller data sets. Checking ancestors for blocked seeds generally yields higher SSS. However, the improvement is at most 0.1% for Twitter and Last.fm. For DBLP, the improvement can be as high as 0.8%. This relatively small improvement is unexpected because checking ancestors for blocked seeds should improve the accuracy of eSSS. However, with small hop

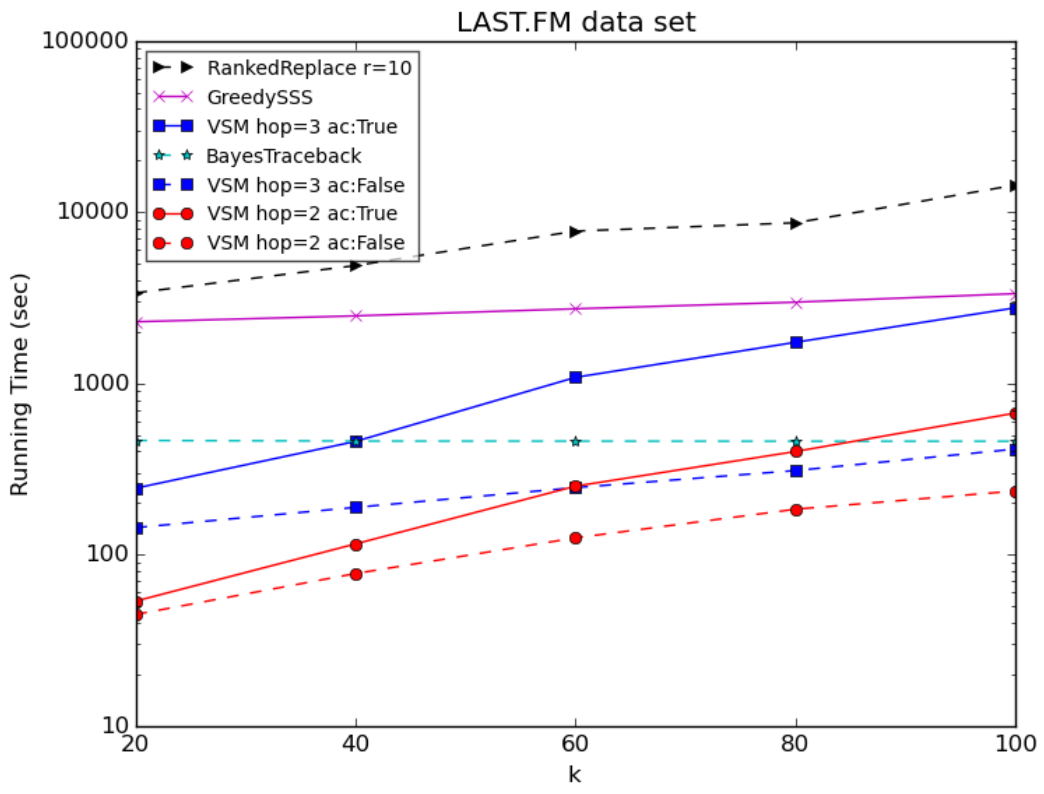


Figure 3.4: Efficiency versus $k$ on LASTFM

limits, the SSS vectors are less accurate, which might degrade the effectiveness of ancestor checking and adjusting the SSS-vectors for blocked seeds.

Figure 3.3, 3.4, and 3.5 plot the CPU times. Computation grows with $k$ and hop limit. Generally, increasing the hop limit by one could increase the computation by an order of magnitude due to more (positive) entries in the vectors (Table 3.5) governed by out-degrees. Checking ancestors for blocked seeds could be 1 to 4 times slower. We also observe that the number of blocked seeds are generally small (data not shown) and deaggregating/aggregating updated vectors when appropriate, instead of always aggregating vectors, reduces computation (Sec. 3.3.4.1). For $k > 80$ GreedySSS runs faster than VSM on DBLP, and for $k > 100$ it is expected

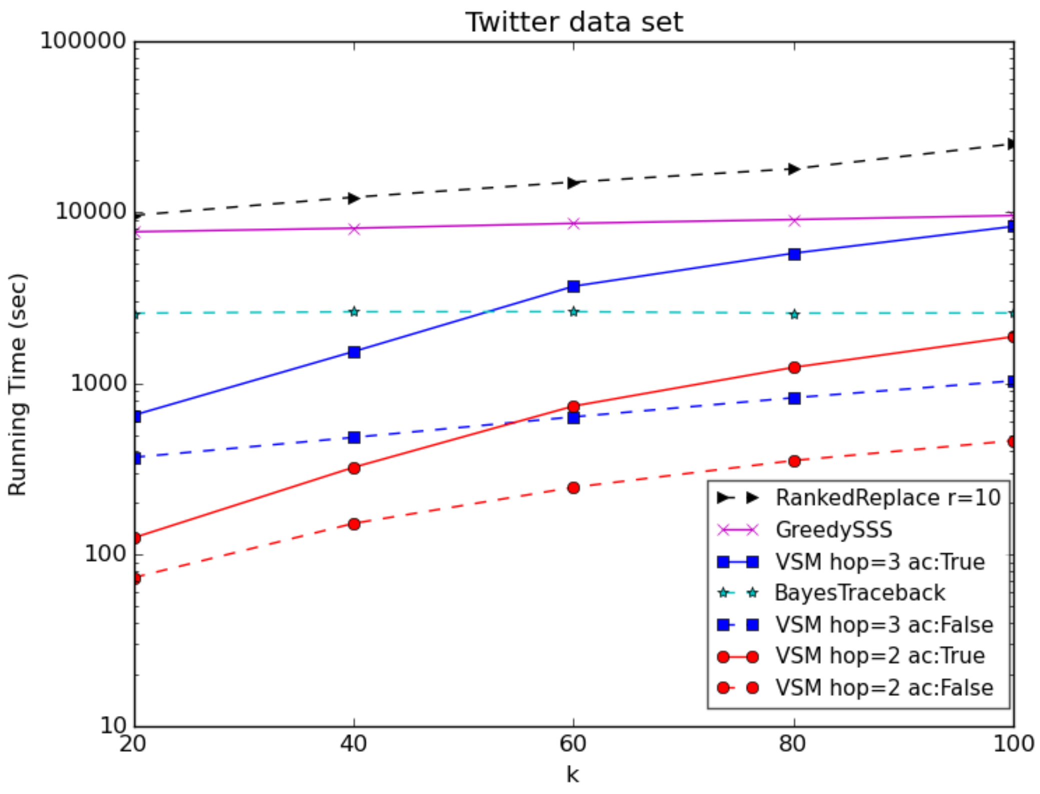


Figure 3.5: Efficiency versus $k$ on Twitter

to be slightly faster on Twitter and LAST.FM. However, its accuracy is consistently lower than VSM on all three data sets.

Table 3.5 displays the number of (positive) entries in SSS vectors that VSM stores for calculating eSSS with $k$=100. The needed memory is less than 1 GB, demonstrating the effectiveness of space reduction in Sec. 3.3.4.1. When the hop limit increases, the number of entries grows rapidly due to the space complexity of $O(b^h)$, where $b$ is the branching factor of a node and $h$ is the hop limit. However, as we discussed above, we do not need a hop limit beyond 3. Table 3.6 displays the number of unique nodes VSM (with ancestor checking) evaluates for the seed set. Generally, VSM evaluates a small number of nodes beyond $k$, which make space

Table 3.4: SSS vs hop limit and ancestor checking

| Hop | AC | $k$=20 | $k$=40 | $k$=60 | $k$=80 | $k$=100 |
|---|---|---|---|---|---|---|
| DBLP | | | | | | |
| 2 | false | 97.7 | 170.7 | 236.5 | 298.0 | 354.8 |
| 3 | false | 97.7 | 170.2 | 236.4 | 298.2 | 355.9 |
| 2 | true | 98.0 | 171.5 | 237.2 | 297.4 | 355.5 |
| 3 | true | 98.2 | 171.5 | 237.3 | 298.7 | 356.0 |
| LAST.FM | | | | | | |
| 2 | false | 571.6 | 822.0 | 1030.8 | 1220.6 | 1388.5 |
| 3 | false | 571.6 | 822.4 | 1033.0 | 1221.5 | 1389.7 |
| 2 | true | 571.6 | 821.9 | 1030.8 | 1220.0 | 1388.5 |
| 3 | true | 571.6 | 822.4 | 1031.6 | 1221.5 | 1390.1 |
| Twitter | | | | | | |
| 2 | false | 315.2 | 489.7 | 625.5 | 743.7 | 848.4 |
| 3 | false | 315.2 | 489.7 | 625.9 | 743.8 | 848.7 |
| 2 | true | 315.3 | 489.7 | 626.0 | 743.7 | 848.7 |
| 3 | true | 315.3 | 489.7 | 626.0 | 744.0 | 849.7 |

Table 3.5: Number of entries ($\times 10^6$) in vectors ($k$=100)

| Hop limit | DBLP | LAST.FM | Twitter |
|---|---|---|---|
| 2 | 0.65 | 1.40 | 2.97 |
| 3 | 5.30 | 10.23 | 26.59 |

reduction in Sec. 3.3.4.1 and pruning in Sec. 3.3.3 effective.

In summary, our results indicate that a hop limit of 2 or 3 and storing vectors for $2k$ nodes are reasonable. However, a hop limit of 2 is preferable to improve speed with a small loss of effectiveness. The additional computation for ancestor checking with a small hop limit might not be worthwhile for some datasets.

### 3.5.8 Accuracy of eSSS in VSM

Table 3.7 shows the error rates of eSSS of the found seed set. When the hop limit increases, the SSS-vectors are more accurate and the error generally decreases. When we check ancestors for blocked seeds, the error generally decreases. Overall,

Table 3.6: Number of unique nodes evaluated for seed set

| Dataset | $k$=20 | $k$=40 | $k$=60 | $k$=80 | $k$=100 |
|---|---|---|---|---|---|
| DBLP | 25 | 48 | 73 | 102 | 133 |
| LAST.FM | 21 | 41 | 68 | 81 | 112 |
| Twitter | 22 | 43 | 66 | 84 | 116 |

Table 3.7: eSSS error in percentage

| Hop | AC | $k$=20 | $k$=40 | $k$=60 | $k$=80 | $k$=100 | Avg of Abs(err) |
|---|---|---|---|---|---|---|---|
| DBLP | | | | | | | |
| 2 | false | -2.14 | -2.71 | -3.03 | -2.58 | -1.88 | 2.47 |
| 3 | false | 1.70 | 2.41 | 2.01 | 1.80 | 1.06 | 1.80 |
| 2 | true | -3.22 | -1.84 | -1.09 | -1.80 | -1.12 | 1.81 |
| 3 | true | -0.17 | -0.04 | -0.13 | 0.13 | 0.11 | 0.12 |
| LAST.FM | | | | | | | |
| 2 | false | -2.44 | -1.67 | -1.23 | -1.45 | -2.11 | 1.78 |
| 3 | false | -0.09 | -0.08 | -0.24 | -0.11 | 0.03 | 0.11 |
| 2 | true | -1.10 | -1.10 | -1.39 | -1.56 | -1.21 | 1.27 |
| 3 | true | -0.06 | -0.08 | -0.05 | -0.21 | -0.09 | 0.10 |
| Twitter | | | | | | | |
| 2 | false | -0.71 | -0.45 | -0.41 | -0.38 | -0.64 | 0.52 |
| 3 | false | 0.33 | 0.08 | 0.21 | 0.21 | 0.27 | 0.22 |
| 2 | true | -0.40 | -0.37 | -0.26 | -0.51 | -0.54 | 0.42 |
| 3 | true | -0.05 | 0.00 | 0.00 | 0.00 | 0.00 | 0.01 |

eSSS with a hop limit of 3 and ancestor checking is within 0.12% of SSS.

## 3.6 Concluding Remarks

We propose estimating SSS (eSSS) from SSS-vectors without running the more expensive *SSS* algorithm (Alg. 1 or 6) in our VSM algorithm. eSSS allows us to efficiently evaluate interactions among seeds and hence effectively select seeds. Also, eSSS is non-negative, monotone, and submodular, which allows VSM to guarantee $(1 - 1/e)$ optimality with respect to eSSS. We further propose considering only the top $2k$ candidates to reduce time and space, without affecting the effectiveness of VSM for the 3 real-world datasets. Our empirical results on the datasets indicate that VSM is slightly more effective ($< 1\%$) than the next most effective algorithm, but significantly more efficient than RankedReplace.

# Chapter 4

# Mining Pros and Cons of Actions from Social Media for Decision Support

## 4.1 Introduction

Discovering pros and cons of actions has many potential applications in decision support, such as purchase recommendation, and finding likely side effects of medications. Establishing a knowledge base on actions and outcomes could assist individuals in making decisions by illustrating potential outcomes given an action that they intend to perform. Those outcomes then can be categorized to form pros and cons of the action.

In this chapter, we propose algorithms to create such a knowledge base on actions and outcomes from social media. Inspired by Kiciman and Richardson [74] we introduce techniques to improve their core components to attain more meaningful pros and cons. Given an action and social media data, our goal is to effectively mine pros and cons of performing the action measured by DCG [69]. Our contributions include:

1. identifying relevant messages containing observations or opinions about the entity of the query by extracting actions and characteristics, as opposed to filtering irrelevant messages in a semi-manual fashion [74],

2. introducing Adjective Vectors to measure semantic similarity between adjectives to improve the clustering quality as in [74],

3. proposing Significance Score (SS) to quantify significance of messages in terms of representing meaningful outcomes, in addition to [74]'s relative likelihood score as a measure to rank distinguishing events, and

4. based on two data sets collected from social media, showing that our algorithm mines more informative pros and cons of the given action compared to [74].

We discuss the related work in Chapter 2. Sec. 5.2 provides the problem statement and describes the different steps of our algorithm. We evaluate our algorithms in Sec. 5.3 and conclude in Sec. 5.4.

## 4.2 Problem Statement and Algorithm

Given an action, the goal is to discover likely outcomes that could be useful in decision making. The inputs include a large corpus of social media messages, and

**Algorithm 7** ProCon($Corpus$, $ActionQuery$)

```
1: find users who performed $ActionQuery$, and collect a timeline of messages for each user from $Corpus$ (Sec. 4.2.1)
2: select messages that express **actions** or **characteristics** (Contribution 1) related to the action query (Sec. 4.2.2)
3: extract events from messages with techniques including **Adjective Vectors** (Contribution 2; Sec. 4.2.3)
4: rank the events via **Significance Score** (**SS**) (Contribution 3; Sec. 4.2.4)
```

a query about performing an action. The output is the most likely outcomes of the query action in the form of a pros-and-cons table. While we maintain the main skeleton of [74], we propose improvements to its core components. The overall algorithm is shown in Alg. 7, and we discuss the steps next.

### 4.2.1 Identifying Relevant Users

From a corpus of social media messages, we find a large number of users who expressed their experience related to the action query. For instance, we search for "adopted a cat|kitty|kitten" when the action query is "adopting a cat". After identifying users who wrote those messages, we collect the entire timeline of messages for each user.

### 4.2.2 Selecting Relevant Messages

The goal of this step is to select messages that describe a relevant situation that the user has experienced. First we use an n-gram approach to filter out non-experiential messages. Then, we select messages with actions and characteristics.

#### 4.2.2.1 Experiential Messages

Relationship between actions and consequences are more meaningful when they are extracted from personal experiences. However, social media messages also include other types, like news and advertisements. We use a simplified unsupervised technique based on n-gram models [74] to filter out undesired messages containing certain keywords and phrases. First, we hand-label a small set of experiential and non-experiential messages (100 messages each). Then, we find a set of n-grams whose likelihood of occurrence in the experiential set is much lower than the other set (we use $n \in \{1, 2, 3\}$ in our experiments). For instance, "We have a kitten ready for adoption" is considered a non-experiential message as it contains "ready for adoption", a trigram with much lower likelihood of occurrence in the experiential set than the other set.

#### 4.2.2.2 Messages with Actions or Characteristics

Actions and characteristics are usually used in natural languages to express effects and outcomes of an action. Hence, we find messages containing actions or characteristics that refer to the *query entity*. Here we define *query entity* as the main object of the query. For example, given query "adopting a cat", the query entity would be "cat". *Actions* and *characteristics* are represented by verbs and adjectives respectively. An *action* is any verb done by or to the query entity, and similarly, a *characteristic* is any adjective mentioned about the query entity. For instance, given action query "Adopting a cat", and sample message "I love coming home and going to bed because my cute cat cuddles with me.", "cuddle" is an action done by the query entity "cat". Also "cute" is a characteristic about the query entity.

Table 4.1: Rules for identification of actions or characteristics

| Type | Rule |
|---|---|
| characteristic | query-entity $e_q$ is subject of verb $v$ & adjective $a$ is adjectival complement of $v$ |
| characteristic | adjective $a$ is adjectival modifier of query-entity $e_q$ |
| action | query-entity $e_q$ is subject of verb $v$ |
| action | query-entity $e_q$ is object of verb $v$ |

To find messages with such grammatical structures, we extract dependency relationships and part of speech tags from each sentence using Stanford Dependency Parser [95]. Then, we find messages with actions or characteristics via a set of handmade grammar rules listed in Table 4.1. The first characteristic rule selects any message where the query entity is subject of a verb having an adjectival phrase. For instance, in "My fat cat is asleep.", "asleep" is the adjectival complement to "is" where "cat" is the subject of "is". The second characteristic rule selects any message with an adjectival modifier for the query entity. For instance, in the previous example, "fat" is an adjectival modifier of the query entity "cat". The first and second action rules select any message where the query entity is a subject or object of a verb. Sec. 4.3.7 shows some examples of messages that are eliminated by our method.

### 4.2.3 Extracting Significant Events

The main goal of this step is to summarize the actions (verbs) and characteristics (adjectives) into events such that each event represents a collection of verbs or adjectives about the same action or characteristic. That is, the verbs and adjectives are clustered (separately) to form events. For example, {"nice", "cute", "lovely"} could form a cluster of adjectives, and {"plays", "runs", "jumps"} could form a cluster of verbs. Event extraction using phrases, as done in previous work [74], provides more effective results than using bag of words as it handles canonicalization. However, it falls short in establishing semantic relationships among words. Since it essentially works based on matching tokens, the clusters are small, with high precision but low recall. As a result, an event can be broken into many small events that otherwise might have formed a significant event. We employ different word representations to establish stronger semantic relationships between verbs and between adjectives. We expect the representations help create clusters with higher recall. Our event extraction algorithm first creates clusters of verbs and adjectives. Then, it identifies a best candidate message and event to represent each cluster.

#### 4.2.3.1 Clustering of Verbs/Adjectives

##### 4.2.3.1.1 Representation of Verbs

We use *Wordnet* [44] hierarchy of verbs. The tree-like hierarchy represents different relation types like *is-a, has-a* and it becomes more specific toward leaves. Thus, a data point in this case is a verb token.

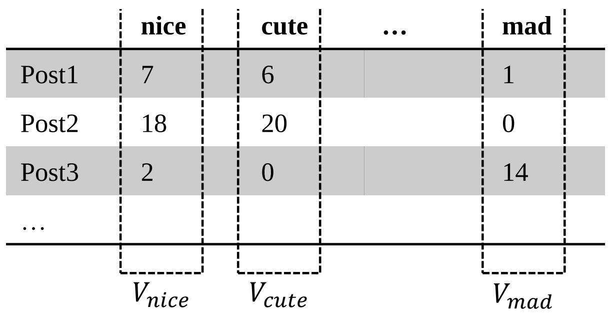
##### 4.2.3.1.2 Representation of Adjectives

Wordnet does not provide a hierarchy for adjectives, and the task of calculating similarity between adjectives remains difficult in the domain. We performed experiments with LESK [87] and Extended LESK [15] algorithms, they perform poorly for clustering. This is mainly because Wordnet provides only limited definitions and relations for adjectives. Hence, we represent adjectives via a different approach.

We propose a different approach based on usage of language by human on the Web where the key assumption is that the query entity likely has a rather unique combination of uncontroversial characteristics. specifically, usually only a small number of characteristics is significantly noticeable about a particular instance of the query entity. Although there are many terms (adjectives) to express one characteristic, these terms are likely to be reused to express other instances with similar characteristic. So, the distribution of similar terms about a characteristic of an entity should be similar.

For instance, each cat has a small number of noticeable characteristics. Characteristics such as “cute”, “sweet”, “nice” and many other terms might be used to describe a cat. Although these words are different, they express the same characteristic of the cat. So, they are likely to co-occur frequently in comments for the same cat. However, it is unlikely to see terms with the opposite meaning (e.g. “ugly”) to describe the same cat. Therefore, co-occurrence of similar terms (e.g. “cute” and “nice”) are likely to be high on “cute” cats. We use this idea to represent adjectives.

We employ social media posts and comments about the query entity to establish semantic relationships between adjectives (in the experiments we use Reddit (www.reddit.com) data for this purpose). Each post is about an instance of the query entity (e.g., a cat), and the comments discuss the same entity from different perspectives, and mostly express similar characteristics with different words. We represent each post, along with its comments, with one vector. Each vector contains the frequency distribution of the adjectives mentioned about the entity in the comments to the post. In other words, each adjective represents a dimension of a post. For example, each row in Fig. 4.1 is a vector representing



| | nice | cute | … | mad |
|---|---|---|---|---|
| Post1 | 7 | 6 | | 1 |
| Post2 | 18 | 20 | | 0 |
| Post3 | 2 | 0 | | 14 |
| … | | | | |

$V_{nice}$
$V_{cute}$
$V_{mad}$

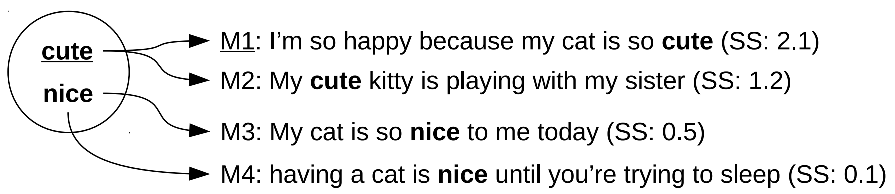
Figure 4.1: Adjective vectors: each number shows the occurrence frequency of an adjective in comments to a social media post.

a post. Then, each adjective is represented by an *adjective vector* containing the frequency of the adjective in all comments to the original posts. Each column in Fig 4.1 illustrates an adjective vector. For example, $V_{nice}$ is the adjective vector for "nice", and $V_{cute}$ is the adjective vector for "cute". Since "nice" and "cute" tend to co-occur more frequently than "nice" and "mad" or "cute" and "mad" for the same cat, $V_{nice} - V_{cute} << V_{nice} - V_{mad}$ is likely to hold. That is, the adjective vectors of similar adjectives are expected to be in close proximity.

In addition to our proposed adjective vectors, we also use GloVe [112], an unsupervised method for creating word embedding trained on different corpora to represent words.

#### 4.2.3.1.3 Distance Function

After representing verbs and adjectives we need a distance function to calculate the distance between a pair of verbs or adjectives. For verbs, we use *hso* [63] which is based on the path length between two nodes of the verb hierarchy. For adjectives, we use cosine measure to calculate distance between adjective vectors or GloVe word vectors.



cute
nice
M1: I'm so happy because my cat is so cute (SS: 2.1)
M2: My cute kitty is playing with my sister (SS: 1.2)
M3: My cat is so nice to me today (SS: 0.5)
M4: having a cat is nice until you're trying to sleep (SS: 0.1)

Figure 4.2: Identifying Representative Messages: $M_1$ is first selected due to its highest Significance Score compared to the other messages. Then, "cute" is identified as the event associated with $M_1$, finally "cute" and $M_1$ are selected as the exemplar to represent the cluster.

#### 4.2.3.1.4 Clustering Algorithm

We use *Kmeans++* [11] to cluster verbs and adjectives separately. As a result we generate a number of action clusters from the action verbs and a number of characteristic clusters from the characteristic adjectives.

### 4.2.3.2 Identifying Representative Messages and Events

The next step is to identify a single significant message that represents each event cluster. First, each cluster member, whether a verb or an adjective, is associated with one or more messages. Next, we use a scoring mechanism that we refer to as *Significance Score* (SS) to pick an exemplar message for the cluster. Then, that message along with the associated verb or adjective represent the event. Fig 4.2 illustrates an example cluster with two adjectives ("cute" and "nice"). Each adjective is associated with all messages that contain the adjective. For example, $M_1$ and $M2$ contain "cute", and $M_3$ and $M_4$ contain "nice". Next, we pick the message with the highest significance. The message (e.g. $M_1$) along with the associated adjective (e.g. "cute") are then selected as the exemplar to represent the cluster.

In order to score messages we employ five factors that contribute to the significance of an event and are sentiment, reasoning, comparison, coverage, and length. We explain each factor in more detail.

##### 4.2.3.2.1 Sentiment Factor

We use VADER [66], a rule-based sentiment model to calculate sentiment factor as an aggregated score normalized between 0 to 1 to show negative to positive respectively.

##### 4.2.3.2.2 Reasoning Factor

Messages with reasoning are highly desired because they provide reasons that probably support their feeling about the outcome. It is calculated as a binary variable that is 1 when any phrase indicating reasoning is observed in the message (e.g. because, therefore, as a result, is why), and it is 0 otherwise.

##### 4.2.3.2.3 Comparison Factor

Comparison is often used in a decision making process. The comparison factor is also calculated as a binary variable that is 1 when comparison tokens are observed and 0 otherwise. The tokens include both keywords and part-of-speech tags. We use Part-of-Speech (POS) tags that are used for comparison words (*JJR, RBR, JJS, RBS*) [71] to identify comparison in sentences. For example, in "cat makes a better pet", the POS tag for "better" is JJR. An advantage of using POS tags is that many words can be represented by one tag. However, POS tagging can be prone to error on incomplete or conversational sentences that usually contain typos. Therefore, we use a small set of keywords as well (*more, most, less, enough*).

#### 4.2.3.2.4 Coverage Factor

A message is a stronger member of its cluster when it contains more than one of the clusters' words. We define the coverage factor as the percentage of cluster words observed in a message normalized between 0 and 1. For example, "my fat cat is asleep" has 0.67 coverage if the cluster contains three adjectives "fat", "asleep" and "tired".

#### 4.2.3.2.5 Length Factor

Length of a message in terms of number of words is another potential indication for a message to be informative. We exclude tokens like urls, hashtags, and mentions. This factor is normalized between 0 and 1 by comparing all messages within a cluster.

We combine the factors via a weighted sum approach:

$$SS = w_{snt}s_{snt} + w_{res}s_{res} + w_{cmp}s_{cmp} + w_{cov}s_{cov} + w_{len}s_{len} \tag{4.1}$$

where $SS$ is the Significance Score, $w_i$ is the weight used for i$^{\text{th}}$ factor, and $s_i$ is the i$^{\text{th}}$ factor. $snt, res, cmp, cov, len$ represent sentiment, reasoning, comparison, coverage, and length factors respectively. Finally, the message with the highest SS and its associated verb or adjective are selected as the event to represent the cluster to which they belong.

Alg. 8 summarizes the steps of extracting significant events. The inputs are the action verbs and characteristic adjectives along with messages to which they belong, the Reddit data discussed in Sec. 4.2.3.1.2, and cluster sizes for verbs events and adjective events. First, the distance matrix for verbs is created using

**Algorithm 8** ExtractSignificantEvents($adjectives, verbs, messages, redditData, k_{vb}, k_{adj}$)

```
1: $distMatrix_{vb} \leftarrow calcDistMatrix(WordnetHierarchy_{vb}, verbs)$
2: $events_{vb} \leftarrow cluster(distMatrix_{vb}, k_{vb})$                    ▷ KMeans++
3: for $event_i \in events_{vb}$ do
4:     for $verb_j \in event_i$ do
5:         $msgList_{vb} \leftarrow getMessages(verb_j, messages)$
6:         $m_{rep} \leftarrow msgMaxSS(msgList_{vb})$
7:         $events_{vb}[i][j] \leftarrow (verb_j, m_{rep})$
8: $adjVectors_{adj} \leftarrow createAdjVectors(adjectives, redditData)$
9: $distMatrix_{adj} \leftarrow calcDistMatrix(adjVectors_{adj}, adjectives)$
10: $events_{adj} \leftarrow cluster(distMatrix_{adj}, k_{adj})$                ▷ KMeans++
11: for $event_i \in events_{adj}$ do
12:     for $adj_j \in event_i$ do
13:         $msgList_{adj} \leftarrow getMessages(adj_j, messages)$
14:         $m_{rep} \leftarrow msgMaxSS(msgList_{adj})$
15:         $events_{adj}[i][j] \leftarrow (adj_j, m_{rep})$
16: return $events_{vb}, events_{adj}$
```

Wordnet hierarchy of verbs (line 1), and then the verbs are clustered to form events (line 2). Next, for each verb in the clusters we get all messages from which the verb was extracted as an action verb. Then, we use SS to to select the best message to represent that verb (line 3-7). Subsequently, we create adjective events. First, we create an Adjective Vector for each adjective using Reddit data, as discussed in Sec. 4.2.3.1.2 (line 8). Then we follow similar steps as we did for the verbs, namely, calculating distance matrix (line 9), clustering (line 10), and selecting a representative for each adjective in the clusters (line 11-15). Finally, the verb events and adjective events are returned. Table. 4.6b in Sec. 4.3.8 exhibits some example clusters with their associated events and representative messages created by Alg. 8.

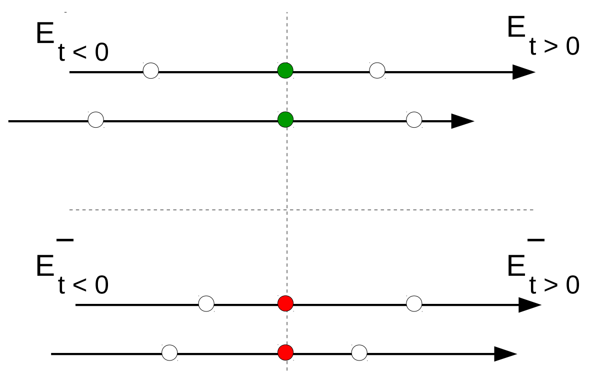


Figure 4.3: Quadrants of user timelines

## 4.2.4 Ranking and Categorizing Events

After extracting significant events and finding a representative message for each event, we rank and categorize them into a pros-and-cons table. We follow three steps to generate the table: First, a collection of highly distinguishing events are selected via correlation analysis used in the previous work citekiciman2015towards. Second, the messages from the first step are ranked by our SS. Third, the ranked messages are categorized into pros or cons when their sentiment scores are large enough. Next, we explain each of the three steps in more detail.

### 4.2.4.1 Distinguishing Events via Correlation

We use correlation analysis on preceding and subsequent events after performing the action to infer potential outcomes. This step is equivalent to the correlational analysis with semantic scoring introduced in the previous work [74]. Distinguishing events are those that occur frequently after the action, but are rarely seen before the action. Additionally, such events are expected to occur rarely after doing the reverse action. In order to perform such correlation analysis we need to temporally align the user timelines such that doing the action or the reverse action occur at $t = 0$ as illustrated in Fig 4.3, the task is to find events that are more likely to occur in one quadrant ($E^+$ for $t > 0$) than in its immediately neighboring quadrants ($E^-$ for $t > 0$ and $E^+$ for $t < 0$). The relative likelihood of an event occurrence in $q$ as compared to $u$ is calculate this as $S_{q,u}(e) = \frac{\tilde{p}_{q,u}(e)}{\hat{p}_u(e)}$ where $\hat{p}_u(e) = \frac{N_q(e)}{|N_q|}$ and $N_q(e)$ is the number of occurrences of an event $e$ in a given quadrant, and $|N_q|$ is the total number of events in a quadrant. Also, $\tilde{p}_{q,u}(e)$ is the Laplace-smoothed probability $\tilde{p}_{q,u}(e) = \frac{N_q(e)+\hat{p}_u(e)m}{|N_q|+m}$.

We apply pair-wise comparison of likelihoods of an event occurrence between the target quadrant $E^+_{t>0}$ and the neighboring quadrants $E^+_{t<0}$ and $E^-_{t>0}$. For an event to be distinguishing the minimum likelihood value between the two comparison should be much greater than one. We use RL (relative likelihood) to select top k% distinguishing events (k=30% in our experiments).

#### 4.2.4.2 Ranking Distinguishing Events

Although distinguishing events ranked by RL are useful, they may not always represent significant events. For example, naming a cat is a distinguishing event, but it is not significant enough to be in the pros-and-cons table. Therefore, we apply our SS to rank events selected from the previous step (Sec. 4.2.3).

#### 4.2.4.3 Categorizing Events

The final step is to categorize the ranked events into two categories of pros and cons. In each iteration, we calculate sentiment score for the next top event and push it into the pros or cons list if the sentiment score condition is met. The pros list accepts events with sentiment score $+0.5$ or larger, and the cons list accepts events with sentiment score $-0.2$ and smaller. We decided the thresholds empirically on a random sample of tweets that do not exist in our main data sets.

**Algorithm 9** RankAndCategorize($events, weights_{SS}, size$
$posSentThr, negSentThr$)

```
 1: $rlScores \leftarrow calcRL(events)$
 2: $topEvents \leftarrow getTopKEvents(events, rlScores)$
 3: $ssScores \leftarrow calcSS(topEvents, weights_{SS})$
 4: $rankedEvents \leftarrow rankWithSS(topEvents, ssScores)$
 5: $pros \leftarrow []; cons \leftarrow []$
 6: repeat
 7:     $e \leftarrow getNextEvent(rankedEvents)$
 8:     $s_e \leftarrow calcSentimentScore(e)$
 9:     if $s_e > posSentThr$ & $len(pros) < size$ then
10:         $pros \leftarrow e$
11:     else if $s_e < negSentThr$ & $len(cons) < size$ then
12:         $cons \leftarrow e$
13: until ($len(pros) = len(cons) = size$)
14: return $pros, cons$
```

The loop stops once both pros and cons lists acquire $m$ events (in our experiments we use $m = 5$).

Alg. 9 summarizes the steps for ranking and categorizing events discussed in Sec 4.2.4. The inputs are the events, including both verb and adjective events, weights to calculate SS, the pros-and-cons table size, and the sentiment thresholds for categorizing events into pros and cons. Each event contains a pair of the word (verb or adjective) and the representative message. First, the top-k events with highest RL scores are selected (line 1-2). Then the top (distinguishing) events are ranked in decreasing order by SS applied to their messages (line 3-4). Next, sentiment score is calculated for each next event's message from the top of the ranked list (line 7-8). If an event's sentiment score falls in the sentiment threshold conditions, it is added to the corresponding list (i.e. pros or cons) (line 9-12). Finally, the pros and cons lists are returned (line 14).

## 4.3 Experimental Evaluation

In this section, we evaluate our algorithm and compare the results with those obtained from the algorithm from Kiciman and Richardson [74] that we call KR15 hereafter.

### 4.3.1 Evaluation Criteria

We evaluate effectiveness of the pros and cons extracted by our technique compared to those of the KR15 algorithm [74]. Each event in the output table consists of an action (verb) or characteristic (adjective) with a representative message. In KR15, each event consists of a phrase and an example representative message. Our evaluation criterion is the extent to which events extracted by each algorithm indicate meaningful pros and cons. To establish the ground truth we asked three evaluators who are graduate students in computer science and engineering fields, and are not authors of this dissertation, to categorize each of the outputted messages into one of three classes {pro, con, neither} based on their personal opinion. We chose an odd number of evaluators to reduce the chance of getting ties. Each message's label was then decided based on the majority of the three opinions. If there is no majority (i.e. each evaluator voted for a different class) we mark that message as neither. To avoid bias toward any algorithm, messages selected by the different algorithms were merged into one set before evaluation.

We measure precision for each generated pros-and-cons table as the percentage of messages identified by the evaluators as either pros or cons. We do not count a message in our precision score if it is in the wrong category. For instance, a pro in the cons list counts as a mistake of the algorithm. Since it would require the evaluators to identify pros and cons from lots of messages from the users, we do not calculate recall.

Moreover, we calculate Discounted Cumulative Gain (DCG) [69] to quantify the ordering quality of the (events) messages in the pros-and-cons table $T$. We expect that more relevant messages appear higher in the table. DCG for a pros-and-cons table is calculated as an average between DCG of pros and cons lists: $DCG_T = \frac{1}{2}\sum_{i=1}^{Pros} \frac{rel_i}{log_2(1+i)} + \frac{1}{2}\sum_{i=1}^{Cons} \frac{rel_i}{log_2(1+i)}$, where $rel_i$ is relevance of the message with rank $i$. Relevance is 1 when $m$ is a pro in the pros list or a con in the cons list. It is 0 otherwise. $DCG_T$ is the overall DCG score of table $T$.

### 4.3.2 Data

We use different social media sources in each part of our system. We discuss each data source next:

#### 4.3.2.1 Twitter

The main source is Twitter where we collect timelines of users who experienced performing the action query. Two data sets were collected for our experiments based on two action queries:

##### 4.3.2.1.1 Cat Adoption

We study the consequences of adopting a cat and collect tweets based on search queries such as “adopted a pet”, “got a pet”, and “got a new pet”, where pet is either “cat”, “kitty” or “kitten”. The query entity is either “cat”, “kitty”, or “kitten”. We expect our algorithm to discover events and tweets that represent the potential outcomes of adopting a cat in the form of pros and cons. We collected

~1.8 million tweets from 980 users who adopted a cat in March and May of 2016. For each user, we collected their timeline from three months before and after adoption.

#### 4.3.2.1.2 Buying iPhone 6

The action query, in this case, is buying an iPhone. We collect tweets based on search queries like “bought an iPhone 6”, “got a new iPhone 6”, where the query entity is iPhone 6. We expect our algorithm to find events and tweets that represent potential outcomes of buying an iPhone 6 cellphone in form of pros and cons. We collected ~2.2 million tweets from 1420 users who purchased an iPhone 6 in January and February of 2017. We collected each user’s timeline from three months before and after they purchased an iPhone 6.

### 4.3.2.2 Reddit

We used Reddit data in form of posts and comments about both query entities (cat and iPhone) to train the Adjective Vectors discussed in 4.2.3.1.2. We collected 400 posts about cats. Each post contains six comments on average. Also we collected 700 posts about iPhone, where each post contains seven comments on average. We only captured comments in the first level. In other words, replies to comments were skipped. Because, our goal is to collect responses to the content in the post, and not those to other comments. Ultimately, we create Adjective Vectors to represent 931 unique adjectives about cat, and 1685 unique adjectives about iPhone.

#### 4.3.2.3 GloVe-Common-Crawl and GloVe-Twitter [1]

In our experiments we also use vectors trained by GloVe [112] as an alternative to our Adjective Vectors. GloVe Common Crawl vectors are 300-dimension, trained based on 840 billion tokens and 2.2 million unique words. GloVe Twitter vectors are 200-dimension, trained on 2 billion tweets, 27 billion tokens and 1.2 million unique words.

### 4.3.3 Procedures

We set up our system with four different models for representation of adjectives and verbs: 1) Our Adjective Vectors trained with the Reddit data set for adjectives and Wordnet hierarchies for verbs, 2) GloVe Common Crawl vectors for adjectives and verbs, 3) GloVe Twitter vectors for adjectives and verbs, 4) GloVe Reddit vectors, where we use our Reddit data set to train 200-dimension vectors by the GloVe algorithm.

Furthermore, we employ a *hybrid* approach, based on voting among the four models. The voting process affects the event ranking component discussed in 4.2.4.2. After selecting the distinguishing events by $RL$, we sort the messages with respect to the number of votes from the four modes in descending order. Next, the messages are selected by SS. The new ordering assigns higher chance of selection to messages with more votes.

We also implemented the algorithm of Kiciman and Richardson [74] (referred to as KR15 in the results) to be able to compare the results. We used Microsoft Web

[1]https://nlp.stanford.edu/projects/glove/

Language Model API[2] to calculate joint probabilities to represent phrases. Then, agglomerative clustering with distance threshold (d=0.75 according to [74]) was applied to create clusters of phrases. Since the original algorithm (KR15) does not mention a specific way to select the representative message for each event, we assume that it picks a message arbitrarily. But we add another version of this algorithm where we use SS to do the selection. We refer to this version as *KR15+SS* in the results. Moreover, we use the following weights for the factors in SS: ($w_{snt} = 1.0$, $w_{res} = 0.67$, $w_{cmp} = 0.67$, $w_{cov} = 0.1$, $w_{len} = 0.1$)

### 4.3.4 Results on Precision

First we compare the effectiveness of our system with four models, our voting-based hybrid approach, GloVe word vectors with three different data sources, KR15, and KR15 with *SS* score (KR15+SS). Table 4.2 illustrates the results. For the Cat data, AdjectiveVectors+Reddit, GloVe+Twitter, and Hybrid outperform the others. GloVe+CommonCrawl and Glove+Reddit show slightly lower performance (80%) than the best algorithms. However, they outperform KR15 and KR15+SS. Hybrid does not produce results better than the individual algorithms. For the iPhone data, AdjectiveVectors+Reddit outperforms all other algorithms. The third column in Table 4.2 shows the average precision on the two data sets. Overall, AdjectiveVectors+Reddit outperforms other algorithms and shows 113% relative improvement over KR15.

[2]https://azure.microsoft.com/en-us/services/cognitive-services/web-language-model/

Table 4.2: Precision of algorithms on the Cat and iPhone6 data; IMP is relative improvement over KR15

| Algorithm | Cat | iPhone6 | Avg | IMP |
|---|---:|---:|---:|---:|
| KR15 | 50% | 30% | 40% | — |
| KR15+SS | 60% | 60% | 60% | 50% |
| AdjectiveVectors+Reddit | **90%** | **80%** | **85%** | **113%** |
| GloVe+CommonCrawl | 80% | 70% | 75% | 88% |
| GloVe+Twitter | **90%** | 50% | 70% | 75% |
| GloVe+Reddit | 80% | 70% | 75% | 88% |
| Hybrid | **90%** | 70% | 80% | 100% |

#### 4.3.4.1 WordVectors+Reddit vs GloVe

AdjectiveVectors+Reddit are more effective than GloVe in general. One reason is GloVe was trained with Twitter or CommonCrawl. The Reddit data contains the same context as the action query (cat adoption or buying iPhone6). In such context, the adjectives are used in similar semantic relationship with the query entity (cat or iPhone6). So, the adjective vectors potentially capture more effective semantic representation of adjectives. However, this is not necessarily the case for context-free data like Twitter and CommonCrawl. As a result, distance measurement between words in context-aware word embedding can be more precise, which leads to higher clustering quality. When we retrain GloVe vectors with our Reddit data the precision increases on average, but it is still lower than AdjectiveVectors+Reddit. One potential reason might be that dimension size of AdjectiveVectors+Reddit is larger than GloVe+Reddit. We tried increasing the vector size for GloVe to match that of AdjectiveVectors+Reddit, but we could not create such vectors due to a bug in the available implementation of GloVe. We intend to investigate this issue and experiment with longer GloVe vectors in future work.

#### 4.3.4.2 AdjectiveVectors+Reddit vs Hybrid

The Hybrid algorithm underperforms our AdjectiveVectors+Reddit on average. A potential reason is that it works well only when most of the individual algorithms perform well. Especially, as seen in the iPhone6 data results, bad messages selected by weak individual algorithms can mislead the Hybrid algorithm by prioritizing those messages. Hybrid has three mistakes for the iPhone data set, but only one of them involves AdjectiveVectors+Reddit. The first mistake is created by votes from AdjectiveVectors+Reddit, GloVe+Reddit, and GloVe+Twitter. The second one is generated by votes from the three GloVe models. The third mistake is produced by votes from GloVe+CommonCrawl and GloVe+Reddit. From these mistakes, we observe that the same messages are selected by different variations of GloVe. One potential reason is that only the training data is different, but the training algorithm (GloVe) and the ranking method (SS) are the same.

#### 4.3.4.3 AdjectiveVectors+Reddit vs KR15/KR15+SS

AdjectiveVectors+Reddit outperforms KR15 because of three reasons: First, it selects messages with actions and characteristics. This is not done in KR15 (discussed in Sec. 4.3.7). Second, it creates clusters of higher quality, mainly because our Adjective Vectors for adjectives and Wordnet for verbs establish stronger semantic relationship between words (discussed in Sec. 4.3.8). Moreover, we use SS to identify the significant representative messages. We also use SS to rank the significant events after selecting the distinguishing ones with RL. In KR15+SS, we use SS to identify representative messages for each event generated by KR15. Although it is helpful, the precision does not increase much. This is likely because events ranked by RL are not as significant as those ranked by SS.

Table 4.3: DCG of the algorithms on the Cat and iPhone6 data set; IMP is relative improvement over KR15

| Algorithm | Cat | iPhone6 | Avg | IMP |
|---|---|---|---|---|
| KR15 | 1.52 | 0.91 | 1.22 | — |
| KR15+SS | 2.08 | 1.93 | 2.01 | 65% |
| AdjectiveVectors+Reddit | 2.64 | **2.42** | **2.53** | **107%** |
| GloVe+CommonCrawl | 2.44 | 1.94 | 2.19 | 80% |
| GloVe+Twitter | **2.71** | 1.52 | 2.12 | 74% |
| GloVe+Reddit | 2.45 | 2.21 | 2.33 | 91% |
| Hybrid | **2.71** | 2.25 | 2.48 | 103% |

#### 4.3.4.4 KR15 vs KR15+SS

In this case the only difference between the two algorithms is that the former uses an arbitrary representative message, but the latter uses SS to do so. The precision increases 20% when we use SS. Although the extracted events and the ranking of those events are the same, SS is able to select better representative messages for the same events.

### 4.3.5 Results on DCG

Table 4.3 shows the ranking effectiveness of pros-and-cons based on DCG. In the Cat data, GloVe+Twitter and Hybrid outperform other models. AdjectiveVectors+Reddit stands second. Moreover, it is observed that Hybrid has picked the best ranking among the individual models. In the iPhone6 data set, AdjectiveVectors+Reddit outperforms other models and shows 107% relative improvement over KR15.

#### 4.3.5.1 AdjectiveVectors+Reddit vs GloVe word vectors

Since our four models (AdjectiveVectors+Reddit, GloVe+CommonCrawl, GloVe+Twitter, GloVe+Reddit) all use SS for both selection of the representative messages and ranking of the events, the difference among their DCG scores is mostly due to the difference in precision.

#### 4.3.5.2 AdjectiveVectors+Reddit vs Hybrid

AdjectiveVectors+Reddit outperforms Hybrid in terms of DCG, on average. However, this is not an effect of ranking, because Hybrid shows lower precision than AdjectiveVectors+Reddit on the iPhone6 data, as shown in Table 4.2.

#### 4.3.5.3 AdjectiveVectors+Reddit vs KR15/KR15+SS

AdjectiveVectors+Reddit outperforms both KR15 and KR15+SS because it uses SS to rank the significant events after selecting the distinguishing ones with RL. The mistakes by AdjectiveVectors+Reddit on the Cat and iPhone6 data occur as high as the second row of cons list. However, KR15 and KR15+SS have more mistakes and they occur as high as the first row.

#### 4.3.5.4 KR15 vs KR15+SS

Although the extracted events and the ranking of those events are the same, our SS finds better representative messages for each event. As a result, many of the mistakes are corrected (20% increase in precision). Since RL is used for ranking of events in KR15+SS, SS can only improve DCG through increasing precision.

### 4.3.6 Examples of Pros-and-Cons tables

#### 4.3.6.1 Cat Adoption

Tables. 4.4a and 4.4b illustrate top five pros and cons generated by KR15 and AdjectiveVectors+Reddit on the Cat data respectively. The events are ranked by RL score in KR15. However, they are ranked by SS in AdjectiveVectors+Reddit. SE represents sentiment score (1=good, -1=bad), and GT illustrates the ground truth based on the majority vote on the message by the evaluators.

Overall, the events extracted by AdjectiveVectors+Reddit represent outcomes of higher quality compared to those extracted by KR15. Although our events are single words (verbs or adjectives), they generally express more meaningful traits of cats compared to the ones extracted by KR15. For instance, if we only use the event column to report as a summary of pros and cons of adopting a cat, the ones reported by AdjectiveVectors+Reddit are more informative than those reported by KR15.

The only mistake from AdjectiveVectors+Reddit occurs in the second row of cons. The reason to select event *"lazy"* as a con goes back to identifying the representative message for a cluster via SS. *"lazy"* belongs to a cluster of three adjectives {*lazy, obese, fat*}. Looking at the messages within the cluster we find one alternative that could have been picked: *"our fat cat had to be put down. He was just in too much pain."* In this case, the associated event would be "fat". The SS values for the message that appears in our cons list and the alternative message are 3.90 and 3.69 respectively. We observe two main reasons for this undesired selection: First, the sentiment scores of the two messages are -0.57 and -0.51 respectively whereas the alternative message conveys much more negative meaning compared to the selected message. Therefore, the sentiment module [66] is not able to evaluate the alternative message effectively. This could be improved by retraining on different data sets, or using more accurate sentiment analysis algorithms. The second reason that the selected message gains higher SS value is that it contains reasoning token *"because"*. Although, the reasoning is correctly identified in the selected message, the alternative message also implies some reasoning that is not captured by SS. In fact, the reasoning token in the alternative message is invisible because the author used two sentences: The first one being the fact, and the second one being the reason. This is a drawback in SS that we aim to address in future work.

#### 4.3.6.2 Buying iPhone6

Tables 4.5a and 4.5b illustrate the top five pros and cons generated by KR15 and AdjectiveVectors+Reddit on the iPhone6 data respectively. The table structure is the same as those for the Cat data.

Overall, the events extracted by AdjectiveVectors+Reddit represent outcomes of higher quality compared to those extracted by KR15. In some cases, the event word (verb or adjective) is not informative if used alone. However, the outcomes can be observed more clearly when the associated representative messages are viewed.

The two mistakes from AdjectiveVectors+Reddit occur in the second and third rows of the pros list. The reason to select the second message in the pros table is its high SS value (4.51). Its sentiment score (0.79) plays an important role in the SS value. But it seems that this large positive score is due to the words (e.g. *"like"*, *"pretty"*) that do not imply any positive sentiment in this message. This

Table 4.4: Example Pros & Cons table generated by algorithms: (a) KR15 on the Cat data, (b) AdjectiveVectors+Reddit on the Cat data. SE represents sentiment score (1=good, -1=bad), and GT (ground truth) illustrates the majority vote on the message by the evaluators (P=pro, C=con, N=neither).

(a) Algorithm: KR15 – Data: Cat

| | Pros | | | | | Cons | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Event | Representative Message | SE | RL | GT | Event | Representative Message | SE | RL | GT |
| 1 | adorable cat | [...] I've adopted an adorable cat within a span of a month. life is great. | 0.81 | 6.2 | P | cat play | My cat doesn't play nice with the dogs [...] so he's in the bedroom for most of the weekend | -0.3 | 6.1 | N |
| 2 | my cat is | my cat is literally curious about anything i eat xd. | 0.63 | 5.9 | P | ignore me | Accidentally punched my cat in the nose. He's going to ignore me and make me feel guilty [...] | -0.63 | 5.5 | C |
| 3 | kitten watched | My kitten watched her namesake get the win and come 1 game away from the. | 0.59 | 5.2 | N | kitten is sad | looking back through the window, it seemed my kitten was sad to see me go to work. | -0.45 | 5.1 | N |
| 4 | wake up | The entire room just screamed "THE HOUND" and my cat didn't wake up so she's super cool. | 0.66 | 4.8 | N | stupid enough | my cat is stupid enough to sleep while eating. | -0.55 | 4.6 | P |
| 5 | my kitten is | My kitten is definitely winning. | 0.53 | 4.3 | P | tearing up | My cat's tearing up my room trying to kill a fly. | -0.69 | 4.1 | C |

(b) Algorithm: AdjectiveVectors+Reddit – Data: Cat

| | Pros | | | | | Cons | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Event | Representative Message | SE | SS | GT | Event | Representative Message | SE | SS | GT |
| 1 | affectionate | Life is better with a cat. Tigger is affectionate and would make a great lap cat. [...] | 0.91 | 4.94 | P | smells | My cat is sleeping in my volleyball bag and I feel bad for him because it smells so bad. | -0.79 | 4.51 | C |
| 2 | sweet | I'm so happy [...] that my sweet kitty came back home to me, I missed you so much sweet girl. | 0.91 | 4.93 | P | lazy | My cat is so lazy he just dragged himself across my bed because he didn't want to get up om*g. | -0.57 | 3.90 | N |
| 3 | cuddles | I love coming home and going to bed because my cat cuddles with me. She is so lovely! | 0.88 | 4.81 | P | ignore | Accidentally punched my cat in the nose. He's going to ignore me and make me feel guilty [...] | -0.77 | 3.52 | C |
| 4 | mews | Aww. My cat mews so cute. I love him so much. | 0.85 | 4.67 | P | claws | My kitten claws my couch and attacks my baby... not so sure I like him anymore. | -0.77 | 3.43 | C |
| 5 | hungry | When my cat is hungry, [...] she just puts on her best Im starving face and stares at me. | 0.53 | 3.80 | P | mad | My cat is so mad at me being that I took her to the vet today. | -0.63 | 2.99 | C |

message contains a reasoning factor (*"because"*), but it is not significant enough to represent an event. The second mistake of AdjectiveVectors+Reddit is the third message in the pros table (event: *restore*) which is selected due to its high SS value. The abbreviation (*"lm*o"*) that indicates humor increases the positive sentiment drastically. However, the sentiment score and SS would be 0.3 and 2.09 respectively without the abbreviation. An alternative message with the next highest SS value (3.89) is *"I've just updated my iphone 6 to allow for native wifi calling but the voice quality vs the app is drastically worse."*. The cluster to which this message belongs contains {*restore, update, reboot, reset*}, and the event in this case would be verb *"update"*. In addition to its large negative sentiment score (-0.57), this message contains a comparison token *"worse"* which is captured by SS because its POS tag is *JJR*. Semantically, it seems that this message could potentially represent a con of buying iPhone6.

#### 4.3.6.3 Cat Adoption vs Buying iPhone6

The task of identifying pros and cons of buying iPhone6 is more difficult than that of cat adoption. Unlike cats, cellphones have many features like screen, battery, apps. Users might express their experience about any individual entity which makes it harder to identify actions and characteristics because they do not directly refer to cellphone. Moreover, sentiment analysis on subjects like cellphone becomes difficult. An off-the-shelf sentiment algorithm calculates the overall sentiment of the sentence. However, the user's sentiment about the entity (e.g., iPhone) or its features might be different than the overall sentiment of the message. This issue has happened in the third pro suggested by AdjectiveVectors+Reddit on the iPhone6 data. The message is quite neutral about iPhone. However, the abbreviation indicates humor that evidently is not related to iPhone. This issue in sentiment analysis potentially decreases ranking performance of our algorithm as the SS mechanism becomes less accurate. We intend to study this issue in future work.

Table 4.5: Example Pros & Cons table generated by algorithms: (a) KR15 on the iPhone6 data, (b) AdjectiveVectors+Reddit on the iPhone6 data. SE represents sentiment score (1=good, -1=bad), and GT (ground truth) illustrates the majority vote on the message by the evaluators (P=pro, C=con, N=neither).

(a) Algorithm: KR15 – Data: iPhone6

| | Pros | | | | | Cons | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Event | Representative Message | SE | RL | GT | Event | Representative Message | SE | RL | GT |
| 1 | my new iphone | I"m in love with my new iPhone 6 Plus | 0.64 | 7.7 | P | iphone charger | listen! if you have an iphone 6 charger [...] i will literally cry because [...] my phone is dead. | -0.83 | 6.9 | N |
| 2 | whip out | gotta whip out the iphone 4 since i got my iphone 6 taken away lmao help me. | 0.77 | 6.8 | N | unlock it | [...] i can give you my iphone 6 and i'll unlock it. | -0.88 | 5.9 | N |
| 3 | got my new | i got my new phone today [...] still on this iphone 6 cuz i haven't ported my number lol. | 0.52 | 6.2 | N | getting my iphone | [...] my sister is 6 and she"s getting my iPhone 6 in two days and has no clue. | -0.78 | 5.3 | C |
| 4 | using my iphone | if y'all text me my phone is restoring rn so i'm using my iphone 6 on wifi lmao hit me on here. | 0.73 | 5.5 | N | had iphone | my brother [...] got the iphone 6 had it for one day broke it and my mom now got him the 7. | -0.68 | 3.4 | N |
| 5 | working per-fectly | got my new iphone 6 working perfectly! | 0.67 | 4.5 | P | second phone | limited budget it's just a second phone for my kink life. i'm currently using an iphone6. | -0.23 | 3.1 | C |

(b) Algorithm: AdjectiveVectors+Reddit – Data: iPhone6

| | Pros | | | | | Cons | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Event | Representative Message | SE | SS | GT | Event | Representative Message | SE | SS | GT |
| 1 | greatest | [...] my iphone 6 plus was the greatest phone i've ever had. [...] | 0.92 | 6.97 | P | trying | hi i'm trying on my iphone6 [...] stuck with an error message at any point i've tried many times! | -0.61 | 4.03 | C |
| 2 | turn on | like my old iphone 6 wouldn't turn on and i'm pretty sure it was bc if the last jailbreak i had,. | 0.79 | 4.51 | N | loses | [...] since updating today my iphone6 loses power rapidly. | -0.61 | 3.99 | C |
| 3 | restore | if y'all text me my phone is restoring rn so i'm using my iphone 6 on wifi lm*o hit me on here. | 0.73 | 4.33 | N | rebooting | my phone on iphone 6 jailbrake just keeps rebooting my phone randomly. so annoying! | -0.58 | 3.85 | C |
| 4 | survived | dropped my naked iPhone 6 into a toilet and it survived so today has been pretty ok. | 0.69 | 4.24 | P | went | my iphone 6 went stupid and i can't get an appt at apple till saturday, [...]. | -0.53 | 3.78 | C |
| 5 | waterproof | my phone just fell in the tub and the music continued to play [...] the iphone6 is waterproof. | 0.64 | 4.06 | P | stupid | i love my apple iphone 6plus. but there is no [...] way i'll spend $1k [...] it's just a stupid phone. | -0.86 | 2.77 | C |

### 4.3.7 Selecting Messages with Actions or Characteristics

Selecting relevant messages (discussed in Sec. 4.2.2.2) brings the focus of our successive components on actions and characteristics about the query entity. Tables 4.4b and 4.5b show some of the selected messages. We eliminate a large number of messages that do not contain observations or opinions about the query entity. Some examples are: *"here is a photo of my cat taken by my friend"*, *"Tips for the first 30 days of cat adoption"*, *"Now tweeting by brand new iPhone6"*, and *"I'm on iPhone 6 in my bed"*.

### 4.3.8 Clustering Quality of AdjectiveVectors+Reddit vs KR15

Tables 4.6a and 4.6b depict examples of clusters including events and messages extracted by KR15 and AdjectiveVectors+Reddit respectively. We observe that our AdjectiveVectors+Reddit is able to create clusters containing multiple words that are semantically homogeneous. However, the clusters created by KR15 are generally smaller and contain phrases that could represent unrelated meanings (e.g. "got my cat", "my cat hobbes"). Therefore, the representative message and the associated event selected by AdjectiveVectors+Reddit is a better representation for the cluster.

## 4.4 Concluding Remarks

We propose actions (verbs) and characteristics (adjectives) to select relevant messages. Also, we propose adjective vectors to represent adjectives and Wordnet

Table 4.6: Examples of clusters and events created by (a) KR15 and (b) AdjectiveVectors+Reddit

(a) KR15

| Cluster | Event | Representative Message |
|---|---|---|
| cat is adorable, adorable cat | adorable cat | [...] I've adopted an adorable cat within a span of a month. life is great. |
| got my cat, my cat hobbes, my cat is, my cat | my cat is | my cat is literally curious about anything i eat xd. |
| stupid enough | stupid enough | my cat is stupid enough to sleep while eating. |

(b) AdjectiveVectors+Reddit

| Cluster | Event | Representative Message |
|---|---|---|
| satanic, annoying, demon, insane, bad, mad, evil, mean | mad | My cat is so mad at me being that I took her to the vet today. |
| gorgeous, sweet, happy, lovely, pretty, sad, friendly | sweet | I am so happy [...] that my sweet kitty came back home to me, [...] |
| bite, pass, grab, claw, cross, hog | claws | My kitten claws my couch and attacks my baby [...] |

entities to represent verbs. As a result, we create clusters of verbs/adjectives of higher quality. We then select a representative message and event for each cluster using SS. We also apply SS in ranking of the events in the final pros-and-cons table. According to precision and DCG on two data sets, the pros and cons discovered by our algorithm are more meaningful than those identified by an existing algorithm (KR15) [74].

# Chapter 5

# Identifying Pros and Cons of Product Aspects Based on Customer Reviews

## 5.1 Introduction

In the recent years, the volume of user-generated social media content has been growing. People across the Web are constantly sharing experiences and opinions about a wide range of situations. Many research lines have focused on using this information as a data source to apply to different domains such as decision support, question answering, machine translation, etc. Reviews of products and services is one of the highly valuable sources of data that can be used to answer questions about the product or service. For example, identifying strengths and weaknesses of a product can be useful for the company that makes the product to improve the weaknesses and add desired features.

In this work we propose an algorithm that extracts product aspects (e.g., camera can be an aspect when the product is a cellphone). and identifies weaknesses and strengths for each aspect in form of pros and cons. We also provide a summary for each product aspect so readers can understand each extracted aspect in a glance before reading the selected pro and con sentences. Our contributions include:

1. jointly identifying product aspects and pros and cons with respect to each aspect,
2. summarizing the aspects in the form of bigrams that show different descriptions or opinions about each product aspect
3. proposing modified Significant Score (SS2) with additional factors to quantify significance of sentences in terms of representing meaningful pros or cons with respect to the product aspects, and
4. based on three data set from Amazon reviews, showing that our algorithm finds pros and cons that are more meaningful and related to the aspects presented in a summarized form compared to [79]

Sec. 5.2 provides the problem statement and describes the different steps of our algorithm. We evaluate our algorithms in Sec. 5.3 and conclude in Sec. 5.4.

## 5.2 Problem Statement and Approach

Given a collection of product reviews, the goal is to automatically extract relevant product aspects and to find the most significant sentences that represent pros and cons for each aspect. The input is a corpus of product reviews, and the output is $M$ product aspects that are often discussed in the reviews, as well as a list of top $K$ pros and cons for each of the aspects. For example if the product is a cell phone, then aspects could be call quality, price, camera, etc.

Our approach has three main steps as shown in Alg. 10. After preprocessing the review text (e.g. removing URLs, emojis and bad characters) and tokenizing them into sentences, the first step is to find the best set of product aspects and assign the review sentences based on the most probable aspects. Second, we employ an scoring method to select messages that are likely to represent pros or cons. We use this score to identify top $K$ sentences for each topic. Then, we use sentiment intensify score with a minimum threshold to separate pros from cons. Third, we summarize product aspects from each topic and present them as bigram phrases using ordered AEMI [77]. We explain each step in more details as follows.

**Algorithm 10** mainProCon

```
Require: $reviews, K, M$
  //1: Extract words from aspects and reviews associated with the aspects:
  asptWds, asptRev = extractAspects($reviews$, $M$)
  //2: Find top K pros and cons sentences for each aspect:
  proSents, conSents = findProsCons(asptWds, asptRev, $K$)
  //3: Summarize aspects in phrases:
  aspPhrases = summarizeAspects(asptWds, proSents, conSents)
  return proSents, conSents, aspPhrases
```

### 5.2.1 Aspect Extraction via Topic Modeling

We use Latent Dirichlet Allocation (LDA) [18] to find topics for two main purposes; First, each topic found by LDA is considered to represent an aspect, where each document is represented by a product review. As a result, two tables are generated by LDA; First, aspect-word where the rows are aspects and the columns are words. Each row contains the probability distribution of words for the corresponding aspect. Second, review-aspect where the rows are reviews and columns are aspects. Each row contains the probability distribution of aspects for the corresponding review. The aspect-word table is used as the source to extract aspect words and aspect bigrams, and the review-aspect table is used to assign reviews to the most probable aspects. Therefore, we expect reviews related to each aspect to be in the same group.

One of the challenges is to find the best number of aspects for any review corpus as it is an input to LDA algorithm. We use a simple optimization method to identify the best number of topics (aspects), where the objective is to minimize the overlap between the aspects found by LDA. The overlap is calculated via a weighted intersection of the first $M$ aspect words of each aspect. The weights are those generated by LDA to show the importance of aspect words within each aspect. We run LDA multiple times by increasing the number of aspects. For each run, we calculate the overlap between each pair of aspects.

After finding the best number of aspects, we calculate the aspect-word and review-aspect matrices. We find the top $M$ important words for each aspect using the aspect-word matrix weights generated by LDA. Moreover, we assign each review to the most probable aspect according to the review-aspect matrix. Alg. 11 illustrates the aspect extraction steps.

**Algorithm 11** extractAspects

```
Require: $reviews, M$
  //1: Find the best number of aspects by minimizing aspect overlap:
  numAspects = minimizeAspectOverlap($reviews$)
  //2: Find aspects with LDA:
  reviewByAspect, aspectByWord = LDA(reviews, numAspects)
  //3: Get the top M words w.r.t the weights in aspectWords matrix:
  aspectWords = getTopWords(aspectByWord, $M$)
  //4: Assign each review to the most probable aspect according to revAspects
  aspectReviews = assignReviews(reviewByAspect)
  return aspectWords, aspectReviews
```

## 5.2.2 Identifying Pros and Cons

The task of identifying pros and cons has four main steps as shown in Alg. 12; First, quantifying co-occurrence via Augmented Expected Mutual Information (AEMI) [77] between aspect words and other words in order to find the representative word pairs involving the aspects. Second, we select sentences that contain at least one representative word pair with positive correlation. Third, we use a scoring method to rank the sentences within each aspect such that those with higher ranks are more likely to be pros or cons. Fourth, we categorize the ranked sentences into pros and cons. Next we explain each step in more details.

### 5.2.2.1 Finding Representative Word Pairs for each Sentence

We propose to find representative word pairs such that each pair includes an aspect word and an arbitrary word. Word pairs tend to be more informative than single aspect words because they add a description or opinion about the aspect word. For example, “water reservoir” explains the function of the aspect word “reservoir”, and “noisy reservoir” specifies an opinion about the aspect word “reservoir”. That is, such word pairs with high co-occurrence are likely representatives of the aspect. Furthermore, sentences that contain such representative word pairs tend to be more significant in terms of representing pros/cons of the product aspects.

We calculate AEMI for all possible pairs of (*word*, *aspectWord*) in the sentences associated to each aspect. *words* can be any word inside the vocabulary constructed from all sentences of an aspect, and *aspectWords* are those from aspect-word table generated by LDA. Equation 5.1 shows AEMI for two words in each pair represented by $A$ and $B$. Each sentence represents an event. $p(A, B)$ is the probability that a sentence contains both $A$ and $B$, and $p(A, \bar{B})$ is the probability that a sentence contains $A$ but does not contain $B$. $p(A)$ represents the probability that a sentence contains $A$, and $p(\bar{A})$ represents the probability that a sentence does not contain $A$.

$$\begin{aligned} AEMI(A,B) = p(A,B)log(\frac{p(A,B)}{p(A)p(B)}) \\ -p(A,\bar{B})log(\frac{p(A,\bar{B})}{p(A)p(\bar{B})}) \\ -p(\bar{A},B)log(\frac{p(\bar{A},B)}{p(\bar{A})p(B)}) \end{aligned} \tag{5.1}$$

Each row of the resulting table is a *word* and each column is an *aspectWord* and each cell contains the AEMI value of the corresponding *word* and *aspectWord*. After calculating the AEMI table, we use it to select sentences that contain word pairs with positive AEMI values because those sentences are much more likely to represent pros/cons related to the product aspects. This way, we exclude sentences that do not contain any representative word pairs. Moreover, we identify the best representative word pair (one with highest AEMI) for each sentence. Such representative word pairs associated to each sentence will be used to calculate significance score in Sec. 5.2.2.2 and summarizing aspects in Sec. 5.2.3.

#### 5.2.2.2 Finding Significant Sentences

We employ Significance Score (SS), a scoring method proposed in[5] and improve it by adding additional factors that can be calculated based on reviews data. By using the new scoring method (SS2) we expect sentences with high scores to have two main characteristics: 1) To find sentences that are likely to represent pros or cons, we use Reasoning, Comparison, Sentiment and Length from the existing factors and add Rating and AEMI as factors. 2) To find sentences that are closely related to the aspect, we introduce Coherence and Coverage as factors.

**Reasoning:** Sentences with reasoning represent pros and cons of higher quality because they provide reasons for the user opinion. Reasoning factor is calculated as a binary variable that is 1 when any phrase indicating reasoning is observed in the sentence (e.g. because, therefore, etc.), and it is 0 otherwise.

**Comparison:** Comparison is often used in expressing pros and cons of a product. Comparison factor is also calculated as a binary variable that is 1 when comparison tokens are observed and 0 otherwise. The tokens include both keywords and part-of-speech tags. We use part-of-speech (POS) tags that represent comparison words (JJR, RBR, JJS, RBS) [71] as well as a small set of keywords (e.g. more, most, less, enough) to reduce the error due to conversational text.

**Length:** Number of words in a message is another indication for a message to be informative. This factor is normalized between 0 and 1 by comparing all sentences within a cluster.

**Sentiment:** We use VADER [47], a rule-based sentiment model to calculate sentiment factor as an aggregated score normalized between -1 and 1 to show negative to positive respectively.

**Coherence:** We add Coherence factor to reward selection of sentences that contain word pairs that tend to co-occur closer within the sentence scope, as such pairs are more likely to be semantically related. Given a (keyword, topical word) pair and the corresponding sentence, coherence score is the distance between the position of the topical word and the keyword in the sentence, normalized by the sentence length.

**Coverage:** Measures what portion of the aspect words that are covered by a given sentence. Therefore, it assigns higher score to sentences that cover more aspect words. We use a weighting mechanism to account for importance of topical words based on the weights in topic-word matrix generated by LDA. Coverage is calculated as sum of the weights of the topical words present in the given sentence, divided by sum of weights of all topical words for the topic.

**Rating:** measures the intensity of the review about the product based on the author's opinion. This is very helpful information as very high and low rating values can indicate high likelihood of positive and negative expressions about the product respectively. The review ratings in our datasets are integers between 1 (negative) and 5 (positive). Thus, we use the following formula to emphasize on very high and low values $score = (rating - 3)^2/4$. By subtracting 3 (the middle point in [1, 5]) from *rating* we calculate how far the rating is from neutral. The result is divided by 4 to normalize the score between 0 and 1. Notice that rating values 1 and 5 both produce the highest value (1) which is desired.

**Relevance:** measures the semantic relevance between the aspect words and each sentence. This factor goes beyond checking for existence of aspect words in each sentence, by measuring the semantic similarity in the word embedding space.

In order to calculate this factor for each sentence, we calculate the cosine similarity between a vector representing the sentence, and another vector representing the $M$ aspect words. We use a pre-trained set of word vectors from GloVe [112] and calculate the vector representations by averaging on the word vectors.

**AEMI:** The AEMI value of the most common bigram of a sentence indicates how often that aspect has been discussed among users. This, we use this value as a factor contributing to our SS2 score.

SS2 score is calculated based on weighted sum of all factors. We specify a weight for each factor to put more emphasize on some factors over others.

$$\begin{aligned} SS2 = {} & w_{res}s_{res} + w_{cmp}s_{cmp} + w_{len}s_{len} + \\ & w_{snt}s_{snt} + w_{coh}s_{coh} + w_{cov}s_{cov} + \\ & w_{rat}s_{rat} + w_{rel}s_{rel} + w_{aemi}s_{aemi} \end{aligned} \tag{5.2}$$

where $w_i$ and $s_i$ are the weight and factor value used for $i^th$ factor, and res, cmp, len, snt, coh, cov, rat, rel, aemi represent reasoning, comparison, length, sentiment, coherence, coverage, rating and AEMI factors respectively. Finally, the sentences associated with each aspect are ordered in decreasing order of their SS2 score and sent to the categorization step.

#### 5.2.2.3 Ranking and Categorizing Sentences

The sentences ordered by SS2 are then categorized into two groups of pros and cons based on their sentiment score. Specifically, first the sentiment score is calculated for each next sentence from the top of the ranked list. If the sentiment score falls in the sentiment threshold conditions the sentence is added to the corresponding list. In our experiments we used threshold of +0.4 and greater for pros and -0.1

and smaller for cons. The process stops once both lists contain $K$ sentences or when no sentences left. Finally, the pros and cons lists are returned.

**Algorithm 12** findProsCons

```
Require: $aspectWords, aspectReviews, K$
  proSents = conSents = []
  for each $aspect_i$ do
      // Calculate AEMI to find the representative word pairs
      $aemi_i$ = AEMI($aspectWords_i$, $aspectReviews_i$)
      // select sentences that contain representative word pairs
      $sents_i$ = selRepSents($aemi_i$, $aspectReviews_i$)
      // Calculate significance of the sentences with SS2
      $rankedSents_i$ = rankBySS2($sents_i$, $aemi_i$)
      // Categorize the K most significant sentences into pros and cons
      $proSents_i$, $conSents_i$=categorize($rankedSents_i$, $K$)
      proSents.add($proSents_i$)
      conSents.add($conSents_i$)
  return proSents, conSents
```

### 5.2.3 Summarizing Aspect Words

In addition to the pros and cons identified by the last steps, we provide a summarized description of the aspects. Although the word pairs generated in Alg. 12 could be used to describe the aspects, the word pairs might not be maningful. For example, {"reservoir", "water"} is a pair but "water reservoir" would be more meaningful. We did not need to consider the word ordering in Alg. 12 because a descriptive/opinion word might be before or after the aspect word. Therefore, to find meaningful bigrams (not just pairs) we calculate AEMI of both bigrams in each pair (e.g. "reservoir water" and "water reservoir"). That is, we calculate an ordered AEMI table for the pairs (*word*, *aspectWord*) to take into account the ordering of the co-occurrence as well. As a result, for a bigram to score high, the two words should often co-occur in the same order. The interpretation of each term in Eq. 5.1, in this case, is different as the probability calculations take the ordering of the word occurrences into account. For example, $p(A, B)$ is the probability that $A$ occurs before $B$ in a sentence. Also, $p(A, \bar{B})$ is the probability that $A$ exists but not followed by $B$, that is, if $B$ occurs before $A$ it is not counted. $p(A)$ is the same as before; the probability that a sentence contains $A$.

After finding the representative bigrams for each aspect, we use a greedy approach to find the minimal set of bigrams that cover the selected pro and con sentences. Specifically, we order the bigrams in decreasing order of their AEMI values, select the first bigram, and eliminate all pro/con sentences that cover (contain) the bigram. The next bigram is selected only if there is a pro/con sentence that is not covered by previously selected bigrams. The process stops once all sentences are eliminated. Finally, the set of selected bigrams is returned. The overall process is illustrated in Alg. 13.

**Algorithm 13** summarizeAspects

```
Require: $aspectWords, aspectReviews, proSents, conSents$
  aspectPhrases = []
  for each $aspect_i$ do
      //Calculate AEMI to find the representative bigrams
      $oaemi_i$ = OrderedAEMI($aspectWords_i$, $aspectReviews_i$)
      //Select the most representative bigrams
      $bigrams_i$ = selTopBigrams($oaemi_i$)
      //find the minimal set of bigrams to cover pro/conSents
      $minBigrams_i$ = findMinSet($bigrams_i$, $proSents$, $conSents$)
      aspectPhrases.add($minBigrams_i$)
  return aspectPhrases
```

## 5.3 Experimental Evaluation

### 5.3.1 Evaluation Criteria

We evaluate the effectiveness of pros and cons extracted by our technique compared to those of Kim and Hovy [79] that we call KH06 hereafter. Our evaluation criterion is the extent to which sentences selected by each algorithm indicate meaningful pros and cons. We calculate Discounted Cumulative Gain (DCG) to quantify the ordering quality of the sentences in the pros-and-cons table. We expect more relevant sentences appear higher in the table. DCG for a pros-and-cons table is calculated as an average between DCG of pros and cons lists. $DCG(sentenceList) = \sum(\frac{rel_i}{log(i+1)})$ where $sentenceList$ can be $prosList$ or $consList$, and $rel_i$ is relevance of sentence $i$.

The evaluation is performed at three levels: 1) Product Level: The pros and cons are considered relevant as long as they are about the product. 2) Aspect Level: The pros and cons should be related to the product and the given aspect. 3) Aspect Summarization: We measure the quality of the representative bigrams extracted by our aspect summarization algorithm. We use DCG to evaluate the quality of the pros and cons extracted by the algorithms in product level and aspect level. Plus, we use precision to evaluate the aspect summarization algorithm. Precision is calculated as the number of correctly identified representative bigrams divided by the total number of identified representative bigrams.

To establish the ground truth of the relevance of a sentence, we asked three evaluators who are graduate students in computer science and engineering fields, and are not authors of this chapter, to label each of the outputted sentences by one of three classes pro, con, neither based on their personal opinion. The relevance of a sentence is one when the predicted class label agrees with the majority of the three opinions. If a majority vote cannot be established the sentence is considered as “neither”. To avoid bias toward any algorithm, messages selected by the different algorithms were merged into one set before evaluation.

The evaluators were asked five questions to provide ground truth for three levels of evaluations. The first two questions were about the quality of the pros and cons, and the next three questions were related to the quality of aspect summarization task.

Given a list of top-K (K=10 in this case) pros and cons sentences, the first question aims to determine whether each sentence represented a pro, con, or neither. This question is at the product level and does not ask the evaluators to consider any product aspects. In the second question, given a list of top-K (K=5 in this case) sentences per aspect, the goal is to determine whether each sentence is a pro, con, or neither with respect to the given aspect. Therefore, this question is at the aspect level and the evaluators should consider whether each sentence is relevant to the aspect. For example, “the cellphone takes great photos.” can be marked as a “pro” when the given aspect is “camera”. But if the given aspect is “size” then the sentence should be marked as “neither”.

Question three provides a list of aspect words, generated by LDA, for each aspect. Question four provides a list of representative bigrams extracted by Alg. 13 for each aspect. In questions three and four we ask “*Does each of the following lists represent/describe at least one product aspect? yes/no*”. In question five, we provide them with a list of aspect words and the corresponding list of bigrams for comparison. We ask “*Which list is more effective/meaningful to represent one (or more) product aspects? The list on the left, the list on the right, or neither*”. We establish the ground truth based on the majority votes from our three evaluators. We break the ties by marking them as "no" for questions three and four, and "neither" for question five to represent a mistake (a false accept). We calculate precision based on the number of lists that contain product aspects according to the evaluators. For the fifth question, we calculate the percentage of the time the evaluators preferred the representative bigrams over the aspect words.

### 5.3.2 Data

We used three data sets from Amazon Reviews data collected and published by McAuley [130, 97]:

- Cellphone data set: contains 837 reviews about a 5.0 inch Android smart phone with product ID (asin) "B0090AAOUW".
- Phone Case data set: contains 766 reviews about a Otterbox Defender Series Case for iPhone 4 & 4S with product ID (asin) "B005SUHPO6".
- Coffee Maker data set: contains 985 reviews about a Keurig k-cup coffee maker with product ID (asin) "B000AQPMHA".

Among the attributes available for each review document, we used *reviewText* to obtain the main body of the review and *overall* to obtain the overall rating of the product assigned by the user between 1 and 5. We removed the non-English tokens, URLs and emojis from the text.

We used Subjective Clues [114] to calculate the opinion-bearing word features for KH06 algorithm. Also we used the pros and cons data set created and used in [91] to train the KH06 algorithm.

### 5.3.3 Procedures

Since KH06 does not propose a solution to extract aspects, we perform the comparison between our algorithm and KH06 in two modes; product-level and aspect-level discussed as follows.

#### 5.3.3.1 Product-Level

We simplify our algorithm to extract product-level pros and cons. Therefore, the aspect extraction step is removed from our algorithm. We refer to this version of our algorithm as "ProCon-PL". The ground truth label for each sentence in this mode is obtained from the first question from the evaluators as discussed in 5.3.1. We used the labeled data set created by Liu et al. [91] to train the KH06 algorithm. The labels in the data set are either "pro" or "con". Next, we used the Amazon reviews data as the test data set. We collected top 10 pros and cons predicted by the algorithm in decreasing order of the prediction values (between 0 and 1) generated by the model.

#### 5.3.3.2 Aspect-Level

In order to add aspects to KH06, we first extract aspects by our aspect extraction Alg. 11. Then for each aspect and its associated sentences, we run KH06 to find top-K pros and cons (K=5 in this case). We refer to this version of KH06 as "KH06-AL".The ground truth for each sentence in this mode is obtained from the second question from the evaluators as discussed in 5.3.1.

Table 5.1: DCG Results at Product-Level

| Cellphone | | | |
|---|---|---|---|
| | Pros | Cons | Avg |
| ProCon-PL | 15.09 | 13.13 | 14.11 |
| KH06 | 14.05 | 10.57 | 12.31 |
| Phone Case | | | |
| ProCon-PL | 14.05 | 10.34 | 12.19 |
| KH06 | 10.98 | 5.71 | 8.35 |
| Coffee Maker | | | |
| ProCon-PL | 15.09 | 15.09 | 15.09 |
| KH06 | 15.09 | 1.11 | 8.10 |

#### 5.3.3.3 Aspect Summarization

Finally, we evaluate the quality of the summarized aspects generated by Alg. 13. The ground truth is based on the third question from the evaluators as discussed in Sec. 5.3.1.

### 5.3.4 Results at Product-Level

Table 5.1 illustrates the DCG results generated by KH06 compared to the product level version of our algorithm (ProCon-PL) discussed in Sec. 5.3.3.1 on the three data sets explained in Sec. 5.3.2. Overall, ProCon-PL outperforms KH06 in product mode on all three data sets. The “Avg” column shows the average between the DCG results of pros and cons.

### 5.3.5 Results at Aspect-Level

Table 5.2 shows the DCG results generated by KH06 with aspect extraction (KH06-AL) discussed in Sec. 5.3.3.2 and our algorithm (ProCon) on the three data sets explained in Sec. 5.3.2. ProCon significantly outperforms KH06-AL in on all three data sets. The “Avg” column shows the average between the DCG results of pros

Table 5.2: DCG Results at Aspect-Level

| Cellphone | | | |
|---|---|---|---|
| | Pros | Cons | Avg |
| ProCon | 9.03 | 9.38 | 9.21 |
| KH06-AL | 2.75 | 3.13 | 2.94 |
| Phone Case | | | |
| ProCon | 9.09 | 8.55 | 8.82 |
| KH06-AL | 2.68 | 2.67 | 2.67 |
| Coffee Maker | | | |
| ProCon | 8.77 | 8.89 | 8.83 |
| KH06-AL | 2.19 | 1.63 | 1.91 |

and cons.

### 5.3.6 Results on Aspect Summarization

Table 5.3 illustrates a qualitative comparison between aspects represented by single words (aspect words) and those by representative bigrams on the three data sets. The precision values indicate the ratio of aspect words or bigrams that represent informative aspects of the product based on the user evaluation. The precision results show that unlike the aspect words, the bigrams extracted by Alg. 13 are likely to represent a meaningful aspect of the product. Table 5.4 shows a direct comparison between aspects represented by single words (aspect words) and representative bigrams for each data set. The values show the percentage of the bigram aspects that were more meaningful than the single word aspects. We can see that at least 75% of the time the aspects represented by bigrams are more informative.

Table 5.5 shows results on aspect summarization task on Coffee Maker data set. The second column depicts the top aspect words extracted by LDA, and the third column shows the aspect summaries (i.e. the minimum set of representative

Table 5.3: Precision of Aspects: The first column shows the precision of aspect words, and the second column shows the precision of representative bigrams, between zero and one.

| | Aspect Word | Representative Bigram |
|---|---|---|
| Cellphone | 0.43 | 1.0 |
| Phone Case | 0.25 | 0.75 |
| Coffee Maker | 0.60 | 1.0 |

Table 5.4: Aspect Words vs. Bigrams: Shows the percentage of instances where the representative bigrams are more informative.

| | % Bigram is Better than Aspect Word |
|---|---|
| Cellphone | 86 |
| Phone Case | 75 |
| Coffee Maker | 80 |

bigrams) extracted via Alg. 13. Overall, the table shows that our algorithm finds five different aspects for the coffee maker product and provides a summary for each aspect based on the reviews left by the users. Moreover, the aspects that are expressed in bigram form tend to be more meaningful and informative compared to the single words generated by LDA in Alg. 11.

### 5.3.7 Examples of Pros and Cons Tables

Table 5.6 and 5.7 show example pros and cons from one example aspect identified by our algorithm (ProCon) compared to those of KH06 in aspect level (KH06-AL) on Coffee Maker data set. The top-5 aspect words that describe this aspect are *water, unit, tank, reservoir, noise.* By looking at the aspect words, one can realize that the aspect is about the water reservoir, tank and other related functionalities as well as effects such as noise. Therefore, it's desired to find pros and cons that are related to this aspect.

Table 5.5: Aspect Summarization on Coffee Maker Data Set

| AspectNum | Aspect Words | Aspect Summary |
|---|---|---|
| 1 | size, elite, brew, model, mug | cup size, expensive elite, Krups model, easy brew |
| 2 | quality, press, french, milk, tons | coffee quality, french coffee |
| 3 | coffee, cup, machine, make, love | coffee cup, coffee maker |
| 4 | water, unit, tank, reservoir, noise | water reservoir, hot water, water heater, unit noise |
| 5 | months, machine, problem, service, customer | customer service, customer support, common problem |

KH06-AL shows many mistakes mostly because the selected sentences are not related to the aspect. For example the first and second example pros in Table 5.6 are pros, but they are not related to the aspect. Although the algorithm is using the aspect extraction Alg. 11 from our ProCon, it is still making many mistakes. The main reason is that the reviews assigned to aspects can be discussing multiple aspects or no aspects in different sentences. As a result, the sentences under an aspect may not be closely related to the aspect. This is where our SS2 method comes handy because, using its factors, it ensures the selected sentences to be closely related to the aspect in addition to be highly likely to represent pros or cons.

In addition to pros and cons, our ProCon provides a summary of each identified aspects in form of bigrams. The first row in Table 5.7 shows the summary. First, the summary bigrams tend to point to aspects of the product that are very related together. They also indicate the key product aspects that are mentioned in the selected reviews. Therefore, by looking at them, one can quickly learn about the aspect and the ideas described by the selected sentences. Furthermore, in contrast to the unigram aspect words generated by LDA, the bigrams tend to be more meaningful and informative. For example, “unit” in the aspect words indicates a very broad concept, but when it is paired with “noise” the reader gets a better idea about a potential negative effect of the coffee maker.

Although our ProCon can find related pros and cons, there are instances that it makes mistakes. For example, the 4th and 5th pro and the 5th con in Table 5.7. The positive point is that the mistakes tend to occur lower in the list. Similar to the results from other aspects or other data sets, the mistakes tend to be “N” (neither) which mostly occur in complex situations where even though the aspect is discussed, what shapes the positive or negative opinion is not about the aspect. For example, in sentence 5 in cons column, the main reason for coffee losing its taste is the the “thin plastic coffee cup” not the “hot water”. One possible solution would be to improve the Reasoning Factor of SS2 such that it identifies the cause and effect. In this example, the first sentence “The coffee also loses its taste” would be the effect, and “hot water is pouring through thin plastic coffee cups” would be the cause. Identifying causality in general is a difficult task, but perhaps it can be applied to sentences with explicit reasoning like the one discussed.

## 5.4 Conclusion

We extract aspects from product reviews and assign the review sentences to the related aspects. We introduce a modified version of Significance Score (SS2) with additional factors to find sentences that are both likely to represent pros and cons, and closely related to the aspect they belong to. Finally, we provide a a

Table 5.6: KH06-AL: Example Pros & Cons for aspect words (water,unit,tank,reservoir,noise) from Coffee Maker data set

| | Pros | | Cons | |
|---|---|---|---|---|
| | Representative Sentence | GT | Representative Sentence | GT |
| 1 | easy to use and makes great coffee. | N | emailed customer support how to drain the unit for long term storage, was told there is no way. | C |
| 2 | great price, easy to use, wonderful espresso. | N | [...] there is no way to fully drain the coffeemaker of water. | C |
| 3 | the water tank is removable for cleaning and has clearly marked water level indicators. | P | i looked online for ways to drain the maker to get it off of my counter while using my espresso machine. | N |
| 4 | the convenience is great, and the coffee is good . | N | i went to keurig's website and learned that you can not drain the internal water reservoir. | C |
| 5 | this coffee maker makes quick, excellent single cups of coffee , which is great for small households. | N | if you forget to turn the power button on , before adding water and your k-cup , the water will drain into the machine. | C |

summary for each product aspect in form of bigrams such that each bigram shows a description or opinion about the aspect.

Table 5.7: ProCon: Example Pros & Cons for aspect words (water,unit,tank,reservoir,noise) from Coffee Maker data set

| Summary | water reservoir, hot water, water heater, unit noise | | | |
|---|---|---|---|---|
| | Pros | | Cons | |
| | Representative Sentence | GT | Representative Sentence | GT |
| 1 | nice thing [...] it was fast, as it pushed the hot water through with a good amount of pressure. | P | i have had no real problems with my b40 coffee maker until i cleaned the water reservoir. | C |
| 2 | the water reservoir is large enough for may daily coffee consumption. | P | be sure that you do not block the water outlets in the needle or you will get less coffee in the cup. | C |
| 3 | i had emptied the removable water reservoir and easily cleaned the appropriate parts. | P | you have to add more water to the water tank that already has enough to make a cup of coffee. | C |
| 4 | again, an add-in paper filter would be useful to slow the water and create a better brew slurry. | N | It worked fine for two weeks. Then it started making a horrible noise. | C |
| 5 | the energy efficiency paradigm with this coffee maker will likely be similar to that of on-demand water heaters | N | the coffee also loses its taste because hot water is pouring through thin plastic coffee cups | N |

# Chapter 6

# Authentication on the Go: Assessing the Effect of Movement on Mobile Device Keystroke Dynamics

## 6.1 Introduction

Mobile devices have become full computing platforms. The data and services they provide have made protecting them of paramount importance. Most devices use a secret knowledge-based means to protect them, such as a password, PIN, or small sketch (e.g., Android pattern lock). These are appropriate measures for initially protecting the device, but they do not provide protection if the device owner does not use them, or if an attacker gains access to an unlocked device.

Keystroke dynamics, or the way in which a person types, has been suggested as a possible means to improve authentication by allowing it to be both *continuous*, protecting the device even after the initial password has been entered, but also *transparent* in that the user need not be distracted from their main task in order to authenticate regularly [135]. This has the potential to not only provide a higher device security level by continuing to authenticate the user after initial password entry, but also improve usability by removing a potentially disruptive request for repeated authentication.

Many of the existing keystroke dynamics studies have relied on the user typing a fixed word or phrase, such as adding keystroke dynamics to password entry, a practice known as *password hardening* [101], but not on dynamic text that changes from sample to sample. Also, much effort has gone into selecting the "best" classifier or the "best" set of features, with only small changes in the apparent distinctive nature of either.

This chapter presents a keystroke dynamics user study designed to determine whether user typing patterns change enough during movement that it can no longer be used as an authenticator. We found on initial analysis that typing patterns over three positions (sitting, standing and walking) were insufficiently distinct to be used as evidence for authentication. This poor result is due to the additional movement that classification algorithms must overcome while typing. We have developed a phased classification approach, seen in Figure 6.1, that takes advantage of such movement. Our phased approach begins with using gyroscope data gathered at each keypress to determine the user's position (sit, stand, or walk). Next, classification models are created for each of the three positions under study that are then used to classify new data. The work presented here is a feasibility study to determine whether the collected gyroscope data is suitable for determining user position. The main novelty in our work is showing that modeling user typing based on their position improves classification rates over building a single, position-independent model. Our results show an improvement in AUC from 66% to 97% when position is considered before classifying keystroke dynamics data. These results indicate that our phased approach has merit; future work includes simulating the classification model to determine its use in practice.

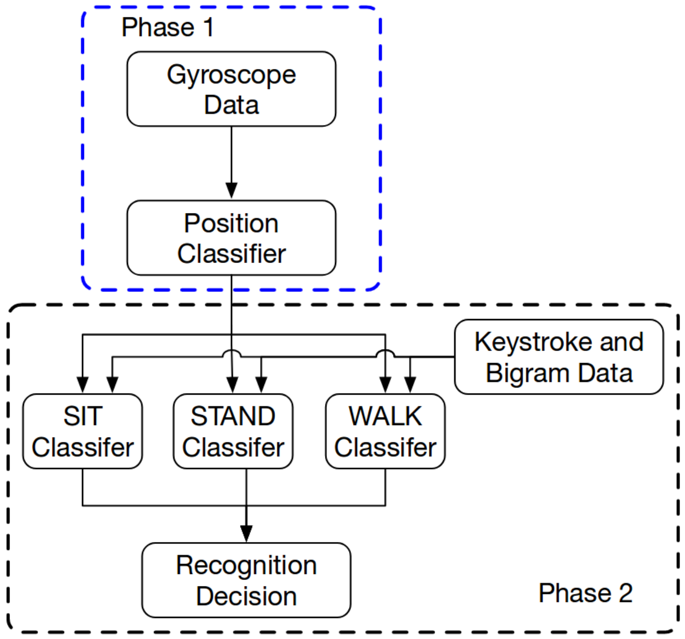


Figure 6.1: Our two-phased classification model

### 6.1.1 Contribution

The major contribution of this chapter is the determination that while typing patterns do change with user movement while typing, our phased classification model allows keystroke dynamics to be used as a viable secondary authentication method under realistic movement and text-acquisition conditions despite typing changes. To our knowledge, this is the first work to create different keystroke models for different user positions; a step that improves the accuracy of user classification. We also provide evidence that gyroscope data gathered at the time of each keypress is suitably distinctive to distinguish between sitting, standing and walking positions. This is significant because gyroscope data is often sampled essentially continuously, which generates a lot of data, uses significant battery power, and requires significant processing in order to be useful. Overall, our results show that keystroke dynamics can be used as secondary or continuous authentication method.

## 6.2 Research Questions

Our research questions are as follows:

1. Does gyroscope data captured at the time keypresses were made provide enough information to tell whether the typist is seated, standing or walking?

2. Does creating multiple position-specific models for a typist provide better classification results compared to using a single model trained on all positions?

3. Does a dynamic text-based system based on the above assumptions provide enough data for a sufficiently distinctive user model?

### 6.2.1 Hypotheses

Based on the above research questions, we present the following hypotheses for our work:

***Hypothesis 1:*** A mobile device user's typing pattern is distinctive enough to use as a secondary or continuous authentication method as determined by achieving an AUC of at least 90%.

***Hypothesis 2:*** Gyroscope data gathered as a key is pressed is distinctive enough to determine whether the typist is seated, standing or walking while typing, as determined by achieving an AUC of at least 90%.

***Hypothesis 3:*** Determining a user's position and classifying based on data from that position only decreases False Accept and False Reject Rates (FAR and FRR, respectively) when compared to classification without determining user position.

We chose 90% as the AUC for Hypothesis 1 in order to justify using our method as a secondary or continuous authentication method, e.g., one that takes place as a supplement to or after primary authentication such as via a password or PIN. This means that near-perfect accuracy is not required, and the balance between FAR and FRR is not as vital as for a primary authentication method. While we may have chosen a lower AUC, we wish to produce a system that may be viable for primary authentication in the future. Therefore, 90% AUC is a value balanced between these two design choices. We chose AUC of 90% for Hypothesis 2 because high accuracy is not required since position determination is not an authentication decision (although it is related to one via our two-phased approach) and thus does not have the security ramifications that characterize authentication decisions.

## 6.3 Threat Model

We assume that an attacker has access to the unlocked mobile device and may have had an opportunity to observe the device owner typing, and thus would know things such as current position, preferred hand position (e.g., index finger, one thumb, both thumbs), device orientation (e.g., landscape or portrait) and a general idea of typing speed. The attacker is assumed to have full knowledge of the biometric authentication system, including all inputs and outputs.

## 6.4 Study and Data Collection

We collected gyroscope data and dynamic text typing data in a user study in a single session. The participant used a custom-built Android app to type phrases provided to them that varied both their position and the device orientation while typing. Specifically, participants were prompted by the experimenter to hold the device in a given orientation (portrait or landscape) and to type while either seated, standing or walking. The participants were told to type as they usually did; specifically, the speed of their typing was not restricted. We also did not provide specific guidance on how to sit, stand or walk. For instance, many participants chose to stand while leaning against a wall, or sit with their arms supported by a table. The only prompts we gave during the experiment were to keep walking if the participant stopped while typing in a walking condition. The study participants filled out a short demographic questionnaire before beginning any typing, and they were allowed to rest between conditions if they wished. Each participant was given the opportunity to practice typing before beginning the first condition; this training data was discarded before analysis. This study was approved by our university's Institutional Review Board (IRB) prior to its start.

### 6.4.1 Participants

We recruited 39 participants (6 female, 33 male) through convenience sampling methods such as personal invitation, emails to mailing lists within our university and word of mouth. The data from three participants was removed from the study due to procedural errors, leaving data from 36 participants (5 female, 31 male). The remainder of this chapter, including study results, refers to the analysis of data from the remaining 36 participants. The average age of participants was 28.3 years (SD = 11.3 years). Participants were not required to have any experience with typing on smartphones, although all participants reported that they owned and used a mobile device, most with soft keyboards. 2 participants were left-handed, and 34 were right-handed. Participant experience on their own mobile device varied: 14 participants used an Android-based device, 18 used an iOS device, 2 used another smartphone, and 2 used a feature (non-smart) phone. 2 participants were considered novices (used their device once a week or less), 3 participants as average (used their device more than once a week but not everyday) and 31 participants as experts (used their device every day or several times each day). Most participants were students, faculty or staff at our university; all participants had at least some post-secondary education, ranging from some undergraduate experience to graduate levels. Participants were not compensated for their participation.

### 6.4.2 Apparatus

We provided each participant with an LG Nexus 5 smartphone for the duration of the experiment. Each device ran Android version 4.4.4 and contained only the

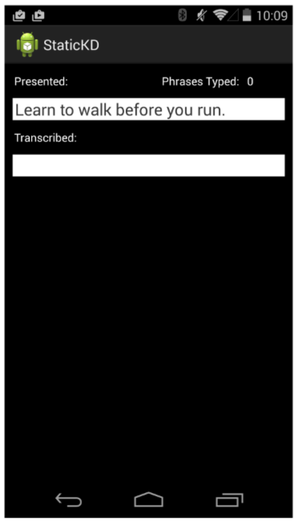


Figure 6.2: Phrase generation app screenshot

standard Android applications. Text entry was facilitated by the use of two bespoke Android applications. The first (see Figure 6.2) displayed the phrase to be typed (non-editable), a text box where the user typed the same phrase, and a counter that displayed the number of phrases the participant had typed in the current experimental condition. This app randomly chose a phrase from a modified version of the standard phrase set provided by MacKenzie and Soukoreff [93] (forthwith called the M&S set); duplicate phrases were permitted.

The second Android app used in this study (see Figure 6.3) was a custom-designed keyboard. It was designed to visually mimic the standard Android keyboard in order to accurately emulate a standard typing environment; the same keyboard design was used by all participants. This app was responsible for gathering the required keystroke and bigram metrics. When the participant pressed a key, the app recorded the key pressed, key hold time, inter-key latency, device orientation, user position and instantaneous gyroscope data (pitch, azimuth and roll). Key hold time is defined as the amount of time that a participant holds down a given key. Inter-key latency is defined as the amount of time between a

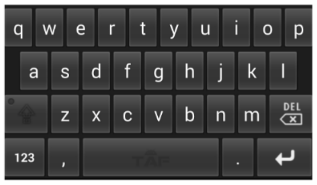


Figure 6.3: Custom keyboard for metric gathering

key up event and the subsequent key down event.

Timing of such typing events is a subject of debate in the keystroke dynamics field [75] as incorrect timing accesses can affect the measured typing pattern of a participant, which in turn has an effect on the reported study results. We mitigated such potential sources of error by using a set of four devices of the same model with the same operating system build, all of which had been reset to factory settings before the experiment began. In addition, we used the same Android applications on each device, and removed the previous participant's gathered data and restarted the application between participants. By using these precautions, we have made all possible efforts to minimize the effects of clock discrepancies on the results of this study.

Our keyboard, which runs as a service on the Nexus 5 devices, replaced the default keyboard in the settings of each device. This design means that when the user tapped on a widget that can accept keystrokes, our keyboard was displayed and subsequently used by the study participants. This allowed for gathering keystrokes from all applications that required typing, meaning that this same keyboard could be used in future work on transparent keystroke dynamics-based authentication.

### 6.4.3 Study Procedure

Each participant first answered a short demographic questionnaire, then was introduced to the app and bespoke soft keyboard they would use for phrase entry. They were given a choice to practice with the standard Android keyboard if they were unfamiliar with it. Most declined as they felt they had enough experience typing on the standard soft keyboard. The participants were allowed to take short breaks after each experimental condition. The participants were instructed to type in their usual manner, and that speed or accuracy were not being measured. They were told that auto-correction and auto-capitalization were disabled, and that if they made mistakes it was their decision whether or not to correct them. The participants were told to not change the device orientation and to remain in the participant mode (sit, stand, walk) they were placed into by the experimenter. They were not told how fast to walk, nor whether they should (or should not) support the device while typing (i.e., leaning against a wall, or with arms supported on a tabletop while seated). No further specifications were given to participants.

Each participant was placed into each of six experimental conditions (see Section 6.4.5 for a of the conditions) by the experimenter and asked to return to the experimenter when they had typed at least 22 phrases (there was a counter at the top of the custom phrase app for this purpose). The number of phrases was chosen in order to gather enough data for analysis, and to provide a similar amount of data for each participant. After completing each condition, the participant was asked to return to the experimenter, who would place the device into the next condition and instruct the participant as to their mode and the device orientation (i.e., "Please type the next set of phrases with the device in portrait while you're seated"). Once the participant had completed each of the six conditions, they were thanked for their time and allowed to leave.

### 6.4.4 Phrase Sets

The stimulus item in this experiment – the prompt that encouraged the user to type – was a randomly selected phrase from a modified set of standard phrases (the M&S set) [93]. Much debate has ensued over the choice of phrases used in text entry experiments. The main issues are that having a non-standard phrase set may impact the results of the study in that using a different phrase sets may result in different experimental results [84]. MacKenzie & Soukoreff addressed the issues of experimental validity (both internal and external) [93], and provided a set of 500 phrases that have been used in various studies. Kristensson & Vertanen opine that the phrase set chosen has an effect on study reproducibility in addition to internal and external validity; it is nearly impossible to reproduce an experiment if the actual phrase set is unknown (i.e., taken from random selections from an unspecified source, such as collecting phrases from "the news") [84].

In choosing a phrase set for this experiment, we kept in mind both internal and external validity as well as study reproducibility, while ensuring our phrase set met the requirements of our experiment. Specifically, we required a phrase set that closely matched letter frequencies in English, was large enough to ensure repeated phrases for the same participant were minimized, and contained upper- and lower-case letters, punctuation and numbers. The M&S set met the first two requirements; but not the last one. To remedy this, we edited the M&S set to have upper-case letters at the start of each phrase, changed text numbers to numeric equivalents (i.e., "eight" was changed to "8"), and added punctuation such as ending periods, exclamation points and question marks, as well as commas where grammatically correct. We believe that doing so created a phrase set that was both ecologically valid and made for a repeatable experiment.

Typically, a true free text typing experiment would require the participant to type whatever came to mind. One issue that arises, though, is what to do when the user cannot think of anything to type since this will affect their standard typing pattern. The role of the phrases in our study were to keep the user typing in as natural a way as possible. Otherwise, the content of the phrases did not have an impact on the accuracy, difficulty, or usability of the typing task. By using phrases rather than free text, however, we have change the user's task from one of creation to one of transcription, which may have an impact on their typing pattern. The advantage we gain is that the typing data is captured with a higher degree of freedom and fewer restrictions than with comparable fixed text experiments, which is arguably more similar to real-world typing situations.

### 6.4.5 Experiment Design and Analysis

Our laboratory-based study used a within-subjects, repeated measures design, in which the study participants were assigned to one of six experimental groups that differed only in the order in which the participant completed each of the six study conditions (see Table 6.1). All participants completed all study conditions. Participants were assigned to each study condition using a 6$x$6 latin squares design in order to minimize learning effects and fatigue. Each session lasted about one hour.

| Study Conditions | | |
|---|---|---|
| **Position** | **Device Orientation** | **Description** |
| Sit | Portrait | **Sitting** in a fixed chair (no casters); Arms optionally **supported** on table; |
| Sit | Landscape | |
| Stand | Portrait | Standing, device **unsupported**; Optionally **leaning** against a wall; |
| Stand | Landscape | |
| Walk | Portrait | **Walking** around a large space, some obstacles; No set speed; most users walked slowly |
| Walk | Landscape | |

Table 6.1: Description of study conditions

### 6.4.6 Data Gathered

The collected keystroke data was sanitized by removing the %, & and $ characters because the experimenter long pressed these keys to indicate a transition between the six study conditions. We used these keys as indicators of a change in device mode between sit, stand and walk. We chose these three keys for this purpose because they did not appear in any of the 500 phrases used as stimulus items, and thus could safely be removed from the dataset without removing valuable user data.

For bigram data, we collected the two characters that make up each bigram (not used during data analysis), the calculated key hold time for that bigram, the device orientation (portrait or landscape) and the participant's position (sit, stand, or walk). We sanitized the data to once again remove the occurrences of the %, &

and \$ characters. In the case of bigrams, we removed the entire bigram from the dataset if any of these three characters appeared as either of the two letters saved. For both keystrokes and bigrams, values greater than 3 SD beyond the mean were considered outliers and removed from the dataset prior to classification.

The gyroscope data was sanitized to remove the occurrences of %, & and \$ but was unchanged otherwise. Since we gathered the gyroscope azimuth, pitch and roll in an instantaneous (i.e., at the moment of a keypress) rather than continuous manner, it was not necessary to window the data into discrete sections, nor to filter the data to remove high- or low-level frequencies as is common in activity recognition studies. Furthermore, since our gyroscope data is not time-scale data since it is discrete rather than continuous measurements, it was not necessary to transform it to the frequency domain before analysis.

We collected a total of 323,064 keystrokes and 289,520 bigrams from all 36 participants, not including practice phrases. The average number of keystrokes gathered per user was 8,974, and the average number of bigrams was 8,042. Since we gathered gyroscope data on each keystroke, we gathered the same amount of gyroscope data as keystrokes with one exception: we did not record instances of using backspace in the keystroke data, but we retained this information for gyroscope data since we were interested in the device movement on each keypress rather than whether that movement was related to a particular key.

### 6.4.7 Feature Vectors

The feature vector for the gyroscope data was simply the $x$, $y$, and $z$ coordinates as gathered during each keypress. The makeup of a feature vector for keystrokes and bigrams, however, is much more complex.

In fixed text keystroke dynamics studies, the feature vector used is quite clear – it is the concatenation of the $n$ key hold times corresponding to the keys pressed when typing the password (often with the ending enter keystroke) and the $n - 1$ inter-key latencies for the associated bigrams. Since all participants type the same password during a study, the feature vectors are the same for each pattern gathered from each participant; the data is complete without missing values. When used outside an experimental setting, the only comparison is between a person's enrolled keystroke metrics when typing their password, and the subsequent keystroke metrics when typing the same password at a later date.

Keystroke biometrics based on dynamic text are more useful when the goal is to gather keystroke information unobtrusively, such as when continuous, transparent authentication is used to verify the identity of a person after initial login. In this situation, we specifically do not want to interrupt the user in what they are doing in order to retype their password, so we instead gather their keystroke metrics as they type as part of their regular device use. We may gather data from them when typing an email, a paper, or a blog post, all of which will have few phrases that appear in all. We gather this data from a custom extension of Android's standard keyboard so that key hold time and inter-key latency, which depend on the keyboard size and key placement, are collected in the same manner for all study participants.

Since dynamic keystroke biometrics cannot depend on getting a fixed amount of text from each participant, nor guarantee that all participants will type the same values, deciding upon the components of the feature vector is a complex task. Intuitively, selecting the most frequently typed characters and bigrams suggests that the most data possible will be retrieved from each participant. However, in practice the most frequently typed characters may vary from person to person. If the most frequently typed English letters are chosen, there might be gaps in our gathered patterns if the participant did not type that letter. This situation gets far worse when considering the frequency of bigrams. These gaps create a much more sparse dataset upon which to base authentication decisions.

One possible solution for this issue is to remove the instances with missing feature values. But it leads to data loss and potential reduce in prediction accuracy. So more data must be gathered to avoid potential loss in prediction accuracy. To handle missing values without a significant data loss, we propose a dynamic feature space where the bigrams and keystrokes involved are those for which we have at least a few instances from the user. For example, early in the data collection process, the classifier may start with a minimum number of bigrams and keystrokes that have been typed thus far (we set this at 4 of each). The feature space then grows as more data is collected. Each new feature, once available, is evaluated based on accuracy on a validation set and added to the current feature space only if the accuracy increases. But even with the minimum number of features, the issue of missing values can occur for some instances, specially for short text in which all features do not appear. To account for this issue, we estimate the missing value based on the average of the available feature values from the same class. For example, say an instance from class "owner" has missing value for bigram "t-h", and we have three values for that bigram {1.5, 1.0, 2.0} in class "owner" and two values {2.2, 3.0} in class "not-owner". The missing value is estimated as 1.5 being the average of the values in "owner" class.

### 6.4.8 Classification

The main goal is to provide decision support for access level control on smartphones via user verification. The access control may apply to different features of the device. For instance, a non-owner user, once identified, might still have access to features like emergency call, checking weather, but they might be denied access to more sensitive features like text messages, contact list, or other apps. Therefore, distinguishing between device owner and others allows the device to make decisions about access control on a process-by-process basis.

We formulate this problem as a two-phase classification approach as displayed in Fig. 6.1. In the first phase, we identify the user position via a classifier trained on the gyroscope data. We use the predicted position to decide which user keystroke model to use in the second phase, where we identify whether the user is the device owner or not.

The typing patterns can vary based on the user position. For example, the typing patterns for a user while seated (i.e. SIT mode) can be quite different with those of the same user while walking (i.e. WALK mode). Moreover, device orientation may change these patterns because the keyboard size is different in landscape and portrait modes. The device orientation can always be identified directly from the device. So, that information is provided along with the test instance. However, information about the user position (sit, stand, or walk) is not explicitly available as part of the test instance. We formulate this problem as a three-class classification task where the goal is to predict device position regardless of identity of the users. We create a position model based on gyroscope data and the feature space discussed in 6.4.7.

We formulate the second phase as two-class classification task, also known as user verification, where the goal is to predict whether the user is the device owner, given user position and device orientation. An alternative setup would be a multi-class classification, also known as user identification, where the goal is to identify each user independently. However, this leads to a more unnecessarily complex issue. Because the number of classes increase with the number of users which potentially reduces classification accuracy. Moreover, in authentication, the device only needs to know the patterns that apply to the device owner, and to distinguish them from the rest of the world. That is, we create a model per user by combining his data with a sample of data from others such that the dataset is balanced with respect to the number of instances per class. Thus, the data contains two classes and the task becomes a binary classification.

To account for the combination of three user positions and two device orientations, we create 6 different models for each user's typing patterns based on the conditions discussed in 6.4.5. The prediction from the classifier in the first phase determines which of the six models should be used. Next, the prediction from the selected model (second phase) determines whether the user is the device owner.

For both classification tasks we experiment with Logistic Regression and Decision Trees. We have three main reasons for choosing these algorithms: First, training time for both of those algorithms is low compared to more complex algorithms such as Artificial Neural Networks (ANN). Moreover, evaluation time for these algorithms is fast which is a plus for the the authentication task. Second, our two-stage classification setup makes each task simpler so that they can be modeled via simpler representations such as a linear model or a single decision tree. Third, both of the algorithms provide transparency for the decisions (predictions) they make. So, one can identify reasons as to why a decision was made by the model. For instance, the coefficients of a linear model are clear indications about importance of each feature. Such information is harder to obtain with more complex algorithms like ANN.

## 6.5 Results and Discussion

We now present our study results and related them back to the hypotheses defined in Section 6.2.1.

### 6.5.1 Position Independent Results

We begin by reporting the results of the naïve method, in which we do not use the gyroscope data to first determine user position. In this case, we mixed data from the sit, stand and walk positions and classified only based on the key hold times and inter-key latencies for two classification algorithms: Decision Tree and Logistic Regression. We chose Decision Tree because of its use in human activity recognition studies [119, 56] and because it is quick to train and classify data. Logistic regression was chosen for its simplicity and ease in understanding feature significance and removing those found to be insignificant. Furthermore, like Decision Tree, logistic regression has a low computation load for training and classifying data, which is an important feature on the constrained memory, power and processing environment on mobile devices. We considered each participant in turn the device owner (their data was considered the positive class), and the other participants as non-owner (their data was considered the negative class). The owner's model was trained on 2/3 of their supplied key hold times (keystrokes) or inter-key latencies (bigrams) plus an equal amount of data randomly selected from the other study participant's data. We used 10-fold cross-validation and report the averages from the 10 folds in Table 6.2. We have reported False Accept Rate (FAR), False Reject Rate (FRR), and the Area Under the Curve (AUC) for the Receiver Operating Characteristic (ROC) curve. We chose to report AUC because it provides, in a single value, the ability of our classifier to distinguish between owner typing patterns and those of others. An AUC value equal to 50% represents a method that is no better than chance; an AUC value equal to 100% is indicative of a perfect classifier.

As can be seen in Table 6.2, the FAR and FRR are very high for both keystrokes and bigrams. For instance, the FAR value of 41.9% for keystroke results using DT indicates that there is a 41.9% probability that an attacker will gain access. This is unacceptably high for any authentication system since it means that nearly half of all attackers will gain access to the mobile device. Similarly, the FRR of 23% for keystrokes using DT represents a nearly one in four likelihood that a legitimate user will be forced to reauthenticate. While reauthentication is less risky in terms of security, it represents an annoyance to users and a reduction in system usability since a legitimate user will have to reauthenticate once out of every four attempts.

The AUC values in Table 6.2 are not much better. Values in the 60-69% range represent a classifier that is only 10-19% better than chance, which is not acceptable even for secondary authentication. Overall, these results indicate that a person's typing pattern changes sufficiently over the three studied positions (sit, stand, walk) that much of the uniqueness in those typing patterns is lost.

Due to these uninspiring results, we chose not to combine key hold time and inter-key latency features as a way of improving classification rates in favor of a potentially better solution: our dual-phased classification model, which is based on first determining the user's position, then classifying using a model built using only user data from that position.

| | **Metric** | **Classifier Metric (%)** | | |
|---|---|---|---|---|
| **DT** | | **FAR** | **FRR** | **AUC** |
| | Keystrokes | 41.9 | 23.0 | 66.9 |
| | Bigrams | 49.3 | 30.5 | 60.3 |
| **LR** | | **FAR** | **FRR** | **AUC** |
| | Keystrokes | 39.0 | 35.6 | 66.2 |
| | Bigrams | 43.3 | 41.7 | 60.7 |

Table 6.2: FAR, FRR and AUC (%) averaged over all participants for keystroke data (key hold time) and bigram data (inter-key latency) using Decision Tree (DT) and Logistic Regression (LR) classifiers. These results do not consider user position (e.g., sit, stand or walk) and are used as a baseline for comparison purposes.

### 6.5.2 Position Dependent Results

The first phase of our two-phased approach is to determine the user's position while they are typing, then classify their typing into owner or not owner based on a model trained only on data from that position. To determine position, we gathered gyroscope data from the mobile device at the moment each key was pressed. Our intuition is that the gyroscopic movement (as measured by the device's pitch, azimuth and roll) will be different when typing while seated, standing or walking. We chose not to measure accelerometer data since it is likely that the accelerometer readings will be different for the walking condition and relatively similar for seated and standing, thus making the latter two positions difficult to distinguish.

#### 6.5.2.1 Gyroscope Data

In order to address Hypothesis 2 regarding the ability of gyroscope data gathered at each keypress to distinguish between the three user positions of sit, stand and walk, we analyzed this data using two classifiers: C4.5 Decision Tree (DT) and Logistic Regression (LR). We used the Weka implementation of these classifiers [62], which were chosen because of their use in activity recognition and keystroke dynamics work, respectively. We used 10-fold cross validation as with the previous classifications.

| | **Pos.** | **Classifier Metric (%)** | | |
|---|---|---|---|---|
| **DT** | | **FAR** | **FRR** | **AUC** |
| | Sit | 4.5 | 10.5 | 97.3 |
| | Stand | 10.3 | 20.2 | 91.5 |
| | Walk | 9.2 | 23.3 | 92.2 |
| **LR** | | **FAR** | **FRR** | **AUC** |
| | Sit | 10.8 | 18.6 | 90.8 |
| | Stand | 15.8 | 39.8 | 82.3 |
| | Walk | 17.7 | 31.1 | 84.5 |

Table 6.3: Gyroscope data FAR, FRR and AUC (%) results averaged over all participants for Decision Tree (DT) and Logistic Regression (LR) classifiers.

As can be seen in Figure 6.3 our results were promising for both DT AND LR, although slightly better for DT. AUC is a valuable measure of classifier accuracy for binary classification problems; Table 6.3 reports the AUC for the position in question considered the positive class, and the other two positions considered the negative class. For example, the AUC of 97.3% for the Sit position for DT is measured based on using Sit as the positive class and Stand and Walk together as the negative class. In general, values of greater than 90% for DT indicate that the gyroscope data gathered is very good at distinguishing between the three user positions. Note that the AUC values for both classifiers for the Sit position are higher than those values for Stand and Walk. We believe this is because users tended to prop their arms on a table while typing during the study, which may mean that the mobile device moved less (or at least differently) compared to the unsupported arm positions while in the Stand and Walk conditions. These promising results show support for accepting Hypothesis 2.

The FRR values in particular, though, are a bit worrisome as they are high for both classifiers. However, these results are not being used to determine authentication suitability, but only to justify using gyroscope data to determine user position. Thus, there is little security risk associated with misclassifying the user's position; such a misclassification simply means the wrong model may be used for classifying keystroke and bigram data. The selection of the wrong model may result in rejecting the legitimate user, which would require reauthentication and thus could affect usability. We intend to explore the impact of such misclassifications in future work.

#### 6.5.2.2 Keystrokes

Once the user's position has been determined, key hold time and inter-key latency data from the user's typing patterns will be classified as owner or not-owner based on three trained models based on data from the three user positions of Sit, Stand and Walk. This section discusses the results of a feasibility study in which the study participants' keystroke and bigram data was classified using position-based models with the DT and LR classifiers to allow for easy comparison to the naïve results shown in Table 6.2.

Table 6.4 shows the FAR, FRR and AUC metrics that result from classifying key hold times over the three user positions. The results for DT for all three metrics are better than those for LR; FAR values for LR in the 18.7% to 20.42% range indicate an unacceptably high one in five chance that an attacker will be mistaken for the legitimate device owner. Furthermore, FRR values of about 23% for LR show a usability problem since nearly one in four authentication attempts by the legitimate owner will fail. Since keystroke dynamics is best used as secondary or continuous authentication method, such a high failure rate is not as great a problem as for primary authentication methods. However, it is still an unacceptably high reauthentication rate. Therefore, we intend to use DT as the classifier of choice in future work.

| | **Position** | **Classifier Metric (%)** | | |
|---|---|---|---|---|
| **DT** | | **FAR** | **FRR** | **AUC** |
| | Sit | 8.5 | 8.4 | 90.3 |
| | Stand | 8.3 | 9.3 | 89.8 |
| | Walk | 7.4 | 8.3 | 91.0 |
| **LR** | | **FAR** | **FRR** | **AUC** |
| | Sit | 18.70 | 23.18 | 82.76 |
| | Stand | 19.63 | 23.73 | 82.32 |
| | Walk | 20.42 | 23.55 | 82.14 |

Table 6.4: Keystroke data (key hold time) FAR, FRR and AUC (%) results averaged over all participants for Decision Tree (DT) and Logistic Regression (LR) classifiers.

#### 6.5.2.3 Bigrams

Previous studies have shown that bigrams on mobile devices are not distinctive as authenticators on mobile devices [117]. However, our results refute this result, perhaps due to the use of position as an initial classification. Table 6.5 shows that bigrams are, in fact, a quite accurate means of authentication. The table shows the results of classifying Sit, Stand and Walk data as separate classification problems; for example, the Sit row for each classifier shows the results of classifying only Sit data into Owner and Not Owner classes; similarly for the Stand and Walk rows.

Table 6.5 shows that the DT classifier outperforms the LR classifier for FRR results, while remaining only slightly higher than LR for FAR values. The AUC values show that inter-key latency is perhaps even slightly more distinctive than key hold time since the bigram AUC values are slightly higher than those of keystrokes. Given that our intent is to use keystroke dynamics for secondary or continuous authentication, AUC values of 89.82% to 93.61% for DT over the three positions are highly encouraging. As with the keystroke data results, we intend to use the DT classifier in future work since the AUC values are comparable to LR, but the FRR values for DT are considerably lower, indicating less likelihood of reauthentication, thereby supporting improved usability.

| | **Position** | **Classifier Metric (%)** | | |
|---|---|---|---|---|
| **DT** | | **FAR** | **FRR** | **AUC** |
| | Sit | 6.9 | 6.0 | 89.8 |
| | Stand | 6.6 | 6.9 | 93.6 |
| | Walk | 6.9 | 7.4 | 92.7 |
| **LR** | | **FAR** | **FRR** | **AUC** |
| | Sit | 5.0 | 13.0 | 92.2 |
| | Stand | 4.3 | 12.7 | 93.1 |
| | Walk | 5.3 | 13.5 | 91.2 |

Table 6.5: Bigram data (inter-key latency) FAR, FRR and AUC (%) results averaged over all participants for Decision Tree (DT) and Logistic Regression (LR) classifiers.

#### 6.5.2.4 Keystrokes + Bigrams

Due to the encouraging keystroke and bigram results after position classification, we combined the key hold time and inter-key latency features while still classifying only one position at a time. Table 6.6 shows the results of this classification; as expected, combining features showed an increase in AUC for both classifiers, although the increase is more notable for the LR classifier. Furthermore, the FAR and FRR values from LR classification are lower for the combined features when compared to those features alone. AUC results of around 97% over all positions for LR move keystroke dynamics into a range we consider suitable for primary authentication, although this must be validated via simulation to determine the impact of battery and processor use, which we leave for future work. Thus, we recommend that keystroke dynamics be used only for secondary or continuous authentication.

| | **Position** | **Classifier Metric (%)** | | |
|---|---|---|---|---|
| **DT** | | **FAR** | **FRR** | **AUC** |
| | Sit | 5.6 | 6.1 | 93.2 |
| | Stand | 6.1 | 5.3 | 93.3 |
| | Walk | 4.8 | 5.6 | 93.9 |
| **LR** | | **FAR** | **FRR** | **AUC** |
| | Sit | 1.7 | 7.0 | 97.3 |
| | Stand | 1.8 | 5.5 | 97.7 |
| | Walk | 1.4 | 6.2 | 97.7 |

Table 6.6: Combination of keystroke (key hold time) and bigram (inter-key latency) data FAR, FRR and AUC (%) results averaged over all participants.

The approximately 90% and up AUC values for DT over keystrokes, bigrams and their combination indicates that using keystroke dynamics as a distinctive information source for authentication is viable, and shows support for accepting Hypothesis 1 of this work.

#### 6.5.2.5 Comparison to Position Independent Results

We now move to comparing the naïve, position independent keystroke and bigram results shown in Table 6.2 to the relevant data in Tables 6.4 and 6.5. The highest AUC for position independent results (Table 6.2) is 66.9% for key hold time data, and 60.7% for inter-key latency data, while the highest AUC values when position is considered are 91.01% for key hold time and 93.61% for inter-key latency. These increases are considerable, and show that considering position before authentication classification is a plausible approach to using keystroke dynamics as a secondary or continuous authentication method. This result is supported by the overall reduction in FAR and FRR values: from lows of 41.5% (FAR) and 23% (FRR) without considering position, to lows of 4.25% (FAR) and 6% (FRR) when position is considered. These reductions indicate that the two-phased approach is better able to minimize both attacker access and reauthentications compared to not considering position. These results show strong support for Hypothesis 3 regarding improvements in classification results when considering device position.

#### 6.5.2.6 Implications

Our threat model outlined in Section 6.3 described possible attacks that can affect the system described in this chapter. In particular, we stated that it is possible that the attacker may observe the device owner typing, and thus may be able to gather information that would allow the attacker to imitate the legitimate device owner. Given that the position-independent results showed us that a user's typing patterns are variable across positions, an attacker would have to learn different typing styles across all positions, which we consider unlikely.

The other implication to consider is what might happen if the first phase of the model (determining position) is incorrect. The effect would be that the wrong model would be used for matching the gathered keystroke information, which may result in rejecting a legitimate user. Given that our method is intended to be used for transparent authentication, there are two possibilities: either that multiple rejected authentication attempts are required to completely block access to the user, which enhances usability since additional user action is not required, but also has serious security ramifications since it increases the possible attack window. The second option is to prompt the device owner to enter a password or PIN when transparent methods are rejected, which has usability implications due to requiring additional user effort, but reduces the possible attack window. The preference for one of these options over the other depends on what type of system it is implemented in; a high-security system may require the latter.

## 6.6 Limitations

As with other user studies, ours has several limitations that must be considered in light of the results provided. Users often walked very slowly during the walking conditions; their focus was on their perceived goal (to enter the phrases) rather than on actually walking. It is likely that in a real-world situation, the user will be intent on walking rather than typing (i.e., if they are running late). Similarly, we observed users propping their arms on a table while typing during the Sit condition, and leaning against a wall during the Stand condition. It is possible that these postures introduced bias in that the static positional data may be more static, thereby further distinguishing this data from that gathered in the Walking condition. This may have resulted in better FAR, FRR and AUC values than in a real-world environment. The phrases themselves may have caused some bias in typing patterns (and removed some ecological validity) as the participant was transcribing the given phrases rather than creating true free text. Furthermore, our study required participants to use an unfamiliar mobile device with an unfamiliar keyboard, which may have had an effect on the participants' typing speed, as well as possibly changing how the keyboard reacts to touch events. We also disabled the predictive and corrective text actions, which affects ecological validity as these are widely used features on soft keyboards. We also did not consider hand postures during our study; participants were permitted to switch between typing with one thumb, both thumbs or any finger while in any of the six experimental conditions. We tested only a small set of classifiers (DT and LR) with few features. Many more possible classifiers exist, including those that take an anomaly detection approach, in which the classifier is trained only on the owner's data rather than adding in some representative negative samples. An anomaly detection approach is considered by some to be more valid for a single-user mobile device as it is unlikely that there will be a significant sample of other people's typing that can be used to create the negative class [23]. While other studies have achieved improved FAR and FRR values by using fused features in a multimodal biometric [23], we chose to use only inter-key latency and key hold time first to conform to other similar studies and also as a minimum baseline result to which future work can be compared. Finally, we collected data in a single session of only one hour in duration, which does not effectively study possible changes in a person's typing pattern over time.

Our final limitation is on the selection of Sit, Stand and Walk as user positions. We chose these based on our intuition that these are the most likely positions in which a user may type. It is unlikely, for instance, that a user will choose to (or be able to effectively) type while running, and positions such as laying down are very similar to both sitting and walking. We plan on addressing this issue with one of two approaches: either create an full activity recognition system that encompasses more positions, or narrowing the positions into those that are similar, such as ambulatory (e.g., walking, running) versus static (e.g., sitting, standing).

While each of these design decisions results in bias that will have differing effects on the results of this study, we believe that the largest effect will be in the overall classification rates, which in the worst case would be artificially high, which would give an inaccurate representation of the predictive power of gyroscope and keystroke data. We note that our results are similar to other studies in this field [116], and plan on removing some of these limitations (particularly those to do with the custom keyboard and disabling predictive and corrective text functions) in the simulation of our phased approach that we mention as future work.

## 6.7 Conclusion

In this chapter we have presented the results of a user study designed to test the efficacy of keystroke dynamics as a potential continuous, transparent authenticator on mobile devices. We first determined via gyroscope data whether the typist was seated, standing or walking, then trained and tested three different models based on dynamic text from each of those three positions. We found that determining position first before classifying typing data resulted in an AUC increase of 30%. Our two-phased model approach of determining position first, then classifying keystroke information thus has merit and should be further examined via simulations. Both our experimental design and our threat model were chosen to provide as much ecological validity as possible given that the study was lab-based. We hope that taking a step back in assessing how much information is required per keystroke, and mimicking how users type in the wild, will provide an important advance in the field of keystroke dynamics.

Overall, our results support continuing research in keystroke dynamics as a transparent authenticator. We removed the need for a feature vector and the associated pre-processing required by them, while supporting a realistic evaluation scenario. We refuted previous results that showed bigram inter-key latency is not as distinctive as hoped for dynamic text, meaning that this feature may now be considered along with key hold time. We also provided support for the idea that transparent authentication may indeed be viable, which may help remove the need for a potentially intrusive and unusable authentication interface.

## 6.8 Future Work

We have begun creating a simulation of the phased approach pictured in Figure 6.1; we will use the simulation to test the effect of the phased approach on device battery and processor consumption, the amount of time needed for a classification decision, and the amount of data needed to reach suitable FAR, FRR and AUC values for continuous authentication. The use of a simulation as a first step will allow us to more closely model real-world typing conditions since our results were from a lab study. With the results of the simulation as a guide, we also plan on creating a prototype of this authentication method for Android devices, which we will test via a longitudinal user study. We will also use the simulation to innovate solutions to the sit-stand confusion, as well as to determine whether a catch-all classifier is suitable for situations where the user is neither sitting, standing nor walking while typing.

# Chapter 7

# Conclusions

We explored different problems related to decision support based on user activities, and introduced new techniques to facilitate the decision making process. We conclude the remarks of each problem as follows.

- In the first problem, we provided technique to improve efficiency of finding a set of top-k users (nodes) that maximize the influence on the network. As a result, not only is our VSM algorithm more effective than the other algorithms that we compared to, it is significantly more efficient as well based on our experiments on three large datasets.

- In the second problem, we propose a technique to identify potential positive and negative outcomes of a given action based on the experience of many users expressed on social media. We identify those outcomes in form of pros and cons with respect to the given action to assist users in decision making process. Next, we evaluate our algorithm on two datasets (two different actions) and show that the pros and cons discovered by our algorithm are more meaningful than those by a state-of-the-art algorithm.

- In the third problem, we introduce a technique to automatically identify product aspects and find pros and cons for each aspect. our algorithm also provides a summary for each identified aspect. We compare the results of our algorithm with the state-of-the-art and show our pros and cons are more meaningful. Our algorithm can be used as a tool to assist users understand strengths and weaknesses of each aspect of a product in buying process. It can also help manufacturers decide which aspect they should improve.

- In the fourth problem, we introduce a two-stage classification method to distinguish between smartphone device owners and others. In the first stage, we determine user position type (SIT, STAND, or WALK). Then, in the second stage, we verify the user based on the keystroke and bigram data collected from their typing patterns. We show that our two-stage method provides high AUC based on a user study on 30 users.

## 7.1 Assumptions and Ethical Issues

Machine learning and data mining algorithms sometimes may seem very attractive to be applied to various real-world problems to help making serious decisions. Although they have been effective in solving many interesting problems like recommender systems, biometrics, robotics, etc., they can be ineffective or even misleading if used in contexts with different assumptions. Like almost all other data mining algorithms, our four solutions are based on many assumptions, and they are designed to be effective only when those assumptions hold. For example, in the problem of identifying the set of top-k nodes to maximize the spread on the network, we have substantial assumptions: For instance, the main assumption in this case is that all messages are considered equally important. Because, the way the weights between the nodes are calculated is based on the frequency of messages sent and received between the two nodes. This method of calculating the weights may be effective in follower/followee or citation networks where the significance of each single message (following or citing someone) can almost be considered as equal. However, calculating the weights based on the frequency seems to be naive when the messages between users consist of written text or spoken conversation. In the second problem, the main assumption is that all users are considered equal or very similar. Essentially, the algorithm assumes that consequences of doing actions are what the majority of people experience when they do the same action. We provide no personalization method to put more emphasize on experience of the users who are more similar to the user who makes the query. Also in the third problem, the assumption is that the product aspects are discussed in the product reviews. So, the output only reflects the aspects discussed by the users. Also, we assume that most of the product aspects can be represented by 2 words (bigrams). However, in some cases more than two words might be needed to express an aspect, for example, "plastic coffee cups" can refer to an internal part of some coffee makers, which could not be expressed as informative with two words. In the fourth problem, the main assumption is that we consider three discrete user positions as walk, sit, and stand. However, in the real life, a user can be in different positions like seated in car or train. So, the three-class model may not be very effective in such situations.

Using these algorithms outside of the context of the assumptions can lead to ethical issues. Because it can lead to making inaccurate or misleading decisions.

To minimize this issue, the only solution seems to be to provide more visibility and caution notes for users in terms of the assumptions build into the algorithms.

## 7.2 Limitations and Future Work

One of the limitations of our algorithms in the second (Chapter 4 and third (Chapter 5) problems is the sentiment intensity calculation which is the reason for many mistaken pros and cons. We used VADER [66], an off-the-shelf algorithm, which is based on general English, not specific subjects. For example, lightness tends to be neutral in general. However, it is considered as a positive description for a cellphone. To address this issue, one idea is to re-train the sentiment algorithm on a dataset that covers the desired domain. For example, user/customer reviews in tech products can be used as training data to provide a more accurate sentiment model when we aim to identify pros and cons of tech products like cellphone.

Another limitation is in our reasoning factor in SS score (Chapter 4) and SS2 score (Chapter 5). The factor is designed to identify sentences that contain reasoning because such sentences have a high potential to represent causes and outcomes. Therefore, those sentences are highly likely to represent pros or cons. Our reasoning factor looks for certain reasoning clues (e.g. because, therefore, etc.) and it fails to find sentences with implicit reasoning. For example in ”My cat mews so cute, I love him so much.”, the reason for loving the cat is because he ”mews so cute”. Identifying causation is a difficult task in general. But it might be simpler in our case because the goal is to only detect it. One potential idea is a supervised learning approach where the feature space for each instance represents a sentence and includes the POS tag of each word. To create feature vectors of equal length, we consider a maximum length in terms of number of words in the sentence. Then shorter sentences are padded with a special POS tag that is reserved for empty words. We can enrich the feature space by using dependency information from grammar dependency trees.

# Bibliography

[1] Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu, and Benyuan Liu. Predicting flu trends using twitter data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pages 702–707. IEEE, 2011.

[2] Leonard M Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266(5187):1021–1024, 1994.

[3] Leonard M Adleman. On constructing a molecular computer. *DNA based computers*, 27:1–21, 1995.

[4] C. Aggarwal, A. Khan, and X. Yan. On flow authority discovery in social networks. In *Proc. SDM*, pages 522–533, 2011.

[5] Ebad Ahmadzadeh and Philip K Chan. Mining pros and cons of actions from social media for decision support. In *Big Data (Big Data), 2017 IEEE International Conference on*, pages 877–882. IEEE, 2017.

[6] Ahmed Awad E Ahmed, Issa Traore, and Almulhem Ahmed. Digital fingerprinting based on keystroke dynamics. In *HAISA*, pages 94–104, 2008.

[7] Jennifer M Allen, Leslie A McFarlin, and Thomas Green. An in-depth look into the text entry user experience on the iphone. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 52, pages 508–512. SAGE Publications Sage CA: Los Angeles, CA, 2008.

[8] Arwa Alsultan and Kevin Warwick. Keystroke dynamics authentication: a survey of free-text methods. *International Journal of Computer Science Issues*, 10(4):1–10, 2013.

[9] Arwa Alsultan and Kevin Warwick. User-friendly free-text keystroke dynamics authentication for practical applications. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 4658–4663. IEEE, 2013.

[10] K. Arnold and J. Gosling. *The Java Programming Language*. Addison-Wesley, Reading, MA, 1996.

[11] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[12] Sitaram Asur and Bernardo A Huberman. Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE, 2010.

[13] Shiri Azenkot and Shumin Zhai. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of the 14th international conference on*

*Human-computer interaction with mobile devices and services*, pages 251–260. ACM, 2012.

[14] Omid Bakhshandeh, Alexis Wellwood, and James Allen. Learning to jointly predict ellipsis and comparison structures. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 62–74, 2016.

[15] Satanjeev Banerjee and Ted Pedersen. An adapted lesk algorithm for word sense disambiguation using wordnet. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 136–145. Springer, 2002.

[16] Timothy Bell, Ian H Witten, and John G Cleary. Modeling for text compression. *ACM Computing Surveys (CSUR)*, 21(4):557–591, 1989.

[17] Francesco Bergadano, Daniele Gunetti, and Claudia Picardi. User authentication through keystroke dynamics. *ACM Transactions on Information and System Security (TISSEC)*, 5(4):367–397, 2002.

[18] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[19] Johan Bollen, Huina Mao, and Xiaojun Zeng. Twitter mood predicts the stock market. *Journal of Computational Science*, 2(1):1–8, 2011.

[20] C. Borgs, M. Brautbar, J. Chayes, and B. Lucier. Maximizing social influence in nearly optimal time. In *Proc. Symp. Discrete Alg.*, pages 946–957, 2014.

[21] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity recognition from accelerometer data on a mobile phone. In *International Work-Conference on Artificial Neural Networks*, pages 796–799. Springer, 2009.

[22] Arnaud Buchoux and N L Clarke. Deployment of keystroke analysis on a smartphone. In *Australian Information Security Management Conference*, page 48, 2008.

[23] Daniel Buschek, Alexander De Luca, and Florian Alt. Improving accuracy, applicability and usability of keystroke biometrics on mobile touchscreen devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 1393–1402. ACM, 2015.

[24] Ting-Yi Chang, Cheng-Jung Tsai, and Jyun-Hao Lin. A graphical-based password keystroke dynamic authentication system for touch screen handheld mobile devices. *Journal of Systems and Software*, 85(5):1157–1165, 2012.

[25] Danqi Chen and Christopher D Manning. A fast and accurate dependency parser using neural networks. In *EMNLP*, pages 740–750, 2014.

[26] Tianyi Chen, Yeliz Yesilada, and Simon Harper. What input errors do you experience? typing and pointing errors of mobile web users. *International journal of human-computer studies*, 68(3):138–157, 2010.

[27] W. Chen, C. Wang, and Y. Wang. Scalable influence maximization for prevalent viral marketing in large-scale social networks. In *Proc. SIGKDD*, pages 1029–1038, 2010.

[28] W. Chen, Y. Wang, and S.Yang. Efficient influence maximization in social networks. In *Proc. SIGKDD*, pages 199–208, 2009.

[29] Wei Chen, Yajun Wang, and Siyu Yang. Efficient influence maximization in social networks. In *Proceedings of the 15th ACM SIGKDD international*

*conference on Knowledge discovery and data mining*, pages 199–208. ACM, 2009.

[30] Hyunyoung Choi and Hal Varian. Predicting the present with google trends. *Economic Record*, 88(s1):2–9, 2012.

[31] Nathan L Clarke and Steven M Furnell. Authenticating mobile phone users using keystroke analysis. *International Journal of Information Security*, 6(1):1–14, 2007.

[32] Nathan Luke Clarke and SM Furnell. Advanced user authentication for mobile devices. *computers & security*, 26(2):109–119, 2007.

[33] NL Clarke, S Karatzouni, and SM Furnell. Transparent facial recognition for mobile devices. In *Proceedings of the 7th Security Conference, Las Vegas, USA, 2nd-3rd June*. Citeseer, 2008.

[34] James Clawson, Thad Starner, Daniel Kohlsdorf, David P Quigley, and Scott Gilliland. Texting while walking: an evaluation of mini-qwerty text input while on-the-go. In *Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services*, pages 339–348. ACM, 2014.

[35] Mauro Conti, Irina Zachia-Zlatea, and Bruno Crispo. Mind how you answer me!: transparently authenticating the user of a smartphone when answering or placing a call. In *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, pages 249–259. ACM, 2011.

[36] Heather Crawford. Keystroke dynamics: Characteristics and opportunities. In *Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on*, pages 205–212. IEEE, 2010.

[37] Heather Crawford, Karen Renaud, and Tim Storer. A framework for continuous, transparent mobile device authentication. *Computers & Security*, 39:127–136, 2013.

[38] Munmun De Choudhury, Scott Counts, and Eric Horvitz. Predicting postpartum changes in emotion and behavior via social media. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3267–3276. ACM, 2013.

[39] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391, 1990.

[40] Reinhard Diestel. Graph theory 3rd ed,. *Graduate texts in mathematics*, 173, 2005.

[41] Paul Dunphy, Andreas P. Heiner, and N. Asokan. A closer look at recognition-based graphical passwords on mobile devices. In *Proceedings of the Sixth Symposium on Usable Privacy and Security*, page 3, 2010.

[42] Ingo Feinerer and Kurt Hornik. *wordnet: WordNet Interface*, 2016. R package version 0.1-11.

[43] Christiane Fellbaum. *WordNet: An Electronic Lexical Database.* Bradford Books, 1998.

[44] Christiane Fellbaum. Wordnet: an eletronic lexical database. cambridge, massachusetts, eua, 1998.

[45] R. Flickenger. *Wireless Hacks*. O'Reilly, 2003.

[46] M. Gast. *802.11 Wireless Networks*. O'Reilly, 2002.

[47] CJ Hutto Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14). Available at (20/04/16) http://comp. social. gatech. edu/papers/icwsm14. vader. hutto. pdf*, 2014.

[48] Cristiano Giuffrida, Kamil Majdanik, Mauro Conti, and Herbert Bos. I sensed it was you: Authenticating mobile users with sensor-enhanced keystroke dynamics. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 92–111, 2014.

[49] Mayank Goel, Leah Findlater, and Jacob O. Wobbrock. Walktype: using accelerometer data to accomodate situational impairments in mobile touch screen text entry. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2687–2696, 2012.

[50] Sharad Goel, Jake M Hofman, Sébastien Lahaie, David M Pennock, and Duncan J Watts. What can search predict. *WWW10*, 2010.

[51] Scott A Golder and Michael W Macy. Diurnal and seasonal mood vary with work, sleep, and daylength across diverse cultures. *Science*, 333(6051):1878–1881, 2011.

[52] A. Goyal, W. Lu, and L. Lakshmanan. CELF++: optimizing the greedy algorithm for influence maximization in social networks. In *Intl. conf. companion on World wide web*, pages 47–48, 2011.

[53] A. Goyal, W. Lu, and L. Lakshmanan. Simpath: An efficient algorithm for influence maximization under the linear threshold model. In *Proc. ICDM*, pages 211–220, 2011.

[54] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete mathematics.* Addison-Wesley, Reading, MA, 1989.

[55] Daniele Gunetti and Claudia Picardi. Keystroke analysis of free text. *ACM Transactions on Information and System Security*, 8(3):312–347, 2005.

[56] Qian Guo, Bin Liu, and Chang Wen Chen. A two-layer and multi-strategy framework for human activity recognition using smartphone. In *Communications (ICC), 2016 IEEE International Conference on*, pages 1–6. IEEE, 2016.

[57] Piyush Gupta and Tim Dallas. Feature selection and activity recognition system using a single triaxial accelerometer. *IEEE Transactions on Biomedical Engineering*, 61(6):1780–1786, 2014.

[58] D. Gurer. Pioneering women in computer science. *Comm. ACM*, 38(1):45–54, 1995.

[59] A. Haeberien, E. Flannery, A. Ladd, A. Rudys, D. Wallach, and L. Kavraki. Practical robust localization over large-scale 802.11 wireless networks. In *Proc. MobiCom*, 2004.

[60] Zhen Hai, Kuiyu Chang, and Jung-jae Kim. Implicit feature identification via co-occurrence association rule mining. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 393–404. Springer, 2011.

[61] Zhen Hai, Gao Cong, Kuiyu Chang, Wenting Liu, and Peng Cheng. Coarse-to-fine review selection via supervised joint aspect and sentiment model. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 617–626. ACM, 2014.

[62] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, 2009.

[63] Graeme Hirst, David St-Onge, et al. Lexical chains as representations of context for the detection and correction of malapropisms. *WordNet: An electronic lexical database*, 305:305–332, 1998.

[64] Minqing Hu and Bing Liu. Mining opinion features in customer reviews. In *AAAI*, volume 4, pages 755–760, 2004.

[65] J Huang, S. Kumar, M. Mitra, W. Zhu, and R. Zabih. Image indexing using color correlogram. *International Journal of Computer Vision*, 35(3):245–268, 1999.

[66] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[67] M. D. Alder J. L. Hutchens. Finding structure via compression. In *Proc. International Conference on Computational Natural Language Learning*, pages 79–82, 1998.

[68] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single- and cross-domain setting with conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1035–1045. Association for Computational Linguistics, 2010.

[69] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[70] Qingye Jiang, Guojie Song, Cong Gao, Yu Wang, Wenjun Si, and Kunqing Xie. Simulated annealing based influence maximization in social networks. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 127–132, 2011.

[71] Nitin Jindal and Bing Liu. Mining comparative sentences and relations. In *AAAI*, volume 22, pages 1331–1336, 2006.

[72] Kyomin Jung, Wooram Heo, and Wei Chen. Irie: Scalable and robust influence maximization in social networks. In *Data Mining (ICDM), 2012 IEEE 12th International Conference on*, pages 918–923. IEEE, 2012.

[73] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *Proc. SIGKDD*, pages 137–146, 2003.

[74] Emre Kıcıman and Matthew Richardson. Towards decision support and goal achievement: Identifying action-outcome relationships from social media. In

*Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 547–556. ACM, 2015.

[75] Kevin Killourhy and Roy Maxion. The effect of clock resolution on keystroke dynamics. In *International Workshop on Recent Advances in Intrusion Detection*, pages 331–350. Springer, 2008.

[76] Kevin S. Killourhy and Roy A. Maxion. Comparing anomaly-detection algorithms for keystroke dynamics. In *2009 IEEE/IFIP International Conference on Dependable Systems Networks*, pages 125–134, 2009.

[77] Hyoung-rae Kim and Philip K Chan. Identifying variable-length meaningful phrases with correlation functions. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 30–38. IEEE, 2004.

[78] J. Kim, S. Kim, and H. Yu. Scalable and parallelizable processing of influence maximization for large-scale social networks. In *Proc. ICDE*, pages 266–277, 2013.

[79] Soo-Min Kim and Eduard Hovy. Automatic identification of pro and con reasons in online reviews. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 483–490. Association for Computational Linguistics, 2006.

[80] M. Kimura, K. Saito, and R. Nakano. Extracting influential nodes for information diffusion on a social network. In *Proc. AAAI*, pages 1371–1376, 2007.

[81] M. Klawe and N. Leveson. Women in computing: Where are we now? *Comm. ACM*, 38(1):29–35, 1995.

[82] D.E. Knuth. Two notes on notation. *Amer. Math. Monthly*, 99:403–422, 1992.

[83] Nozomi Kobayashi, Ryu Iida, Kentaro Inui, and Yuji Matsumoto. Opinion mining on the web by extracting subject-aspect-evaluation relations. In *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*, pages 86–91, 2006.

[84] Per Ola Kristensson and Keith Vertanen. Performance comparisons of phrase sets and presentation styles for text entry evaluations. In *Proceedings of the 2012 ACM international conference on Intelligent User Interfaces*, pages 29–32. ACM, 2012.

[85] Himabindu Lakkaraju, Chiranjib Bhattacharyya, Indrajit Bhattacharya, and Srujana Merugu. Exploiting coherence for the simultaneous discovery of latent facets and associated sentiments. In *Proceedings of the 2011 SIAM international conference on data mining*, pages 498–509. SIAM, 2011.

[86] John J. Leggett, Glen Williams, Mark Usnick, and Mike Longnecker. Dynamic identity verification via keystroke characteristics. *International Journal of Human-computer Studies International Journal of Man-machine Studies*, 35(6):859–870, 1991.

[87] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone. In *Proceedings of the 5th Annual International Conference on Systems Documentation*, SIGDOC '86, pages 24–26, New York, NY, USA, 1986. ACM.

[88] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. Glance. Cost-effective outbreak detection in networks. In *Proc. SIGKDD*, pages 420–429, 2007.

[89] R. Lipton. Using DNA to solve NP-complete problems. *Science*, 268:542–545, 1995.

[90] Richard J Lipton. Speeding up computations via molecular biology. *DNA Based Computers*, 27:67–74, 1995.

[91] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: analyzing and comparing opinions on the web. In *Proceedings of the 14th international conference on World Wide Web*, pages 342–351. ACM, 2005.

[92] Yue Lu, ChengXiang Zhai, and Neel Sundaresan. Rated aspect summarization of short comments. In *Proceedings of the 18th international conference on World wide web*, pages 131–140. ACM, 2009.

[93] I Scott MacKenzie and R William Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI'03 extended abstracts on Human factors in computing systems*, pages 754–755. ACM, 2003.

[94] Emanuele Maiorana, Patrizio Campisi, Noelia Gonzlez-Carballo, and Alessandro Neri. Keystroke dynamics authentication for mobile phones. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 21–26, 2011.

[95] Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[96] Philip Marquardt, Arunabh Verma, Henry Carter, and Patrick Traynor. (sp)iphone: decoding vibrations from nearby keyboards using mobile phone accelerometers. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 551–562, 2011.

[97] Julian McAuley and Alex Yang. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee, 2016.

[98] Arik Messerman, Tarik Mustafic, Seyit Ahmet Camtepe, and Sahin Albayrak. Continuous and non-intrusive identity verification in real-time environments based on free-text keystroke dynamics. In *2011 International Joint Conference on Biometrics (IJCB)*, pages 1–8, 2011.

[99] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[100] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[101] Fabian Monrose, Michael K Reiter, and Susanne Wetzel. Password hardening based on keystroke dynamics. *International Journal of Information Security*, 1(2):69–83, 2002.

[102] Fabian Monrose and Aviel D. Rubin. Keystroke dynamics as a biometric for authentication. *Future Generation Computer Systems*, 16(4):351–359, 2000.

[103] Mark Myslín, Shu-Hong Zhu, Wendy Chapman, and Mike Conway. Using twitter to examine smoking behavior and perceptions of emerging tobacco products. *Journal of medical Internet research*, 15(8):e174, 2013.

[104] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM, 2003.

[105] Mohammad Faizuddin Mohd Noor, Simon Rogers, and John Williamson. Detecting swipe errors on touchscreens using grip modulation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 1909–1920, 2016.

[106] Alexandra Olteanu, Onur Varol, and Emre Kıcıman. Distilling the outcomes of personal experiences: A propensity-scored analysis of social media. In *Proc. of The 20th ACM Conference on Computer-Supported Cooperative Work and Social Computing*, 2017.

[107] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[108] Siddharth Patwardhan. *Incorporating dictionary and corpus information into a context vector measure of semantic relatedness.* PhD thesis, University of Minnesota, Duluth, 2003.

[109] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Using measures of semantic relatedness for word sense disambiguation. In

*International Conference on Intelligent Text Processing and Computational Linguistics*, pages 241–257. Springer, 2003.

[110] Michael J Paul and Mark Dredze. You are what you tweet: Analyzing twitter for public health. *ICWSM*, 20:265–272, 2011.

[111] A. Pearl. Women in computing. *Comm. ACM*, 38(1):26–28, 1995.

[112] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[113] Matthew Richardson. Learning about the world through long-term query logs. *ACM Transactions on the Web (TWEB)*, 2(4):21, 2008.

[114] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112. Association for Computational Linguistics, 2003.

[115] Ellen Riloff, Janyce Wiebe, and Theresa Wilson. Learning subjective nouns using extraction pattern bootstrapping. In *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003-Volume 4*, pages 25–32. Association for Computational Linguistics, 2003.

[116] Chao Shen, Tianwen Yu, Sheng Yuan, Yunpeng Li, and Xiaohong Guan. Performance analysis of motion-sensor behavior for user authentication on smartphones. *Sensors*, 16(3):345, 2016.

[117] Terence Sim and Rajkumar Janakiraman. Are digraphs good for free-text keystroke dynamics? In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[118] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[119] Xing Su, Hanghang Tong, and Ping Ji. Activity recognition with smartphone sensors. *Tsinghua Science and Technology*, 19(3):235–249, 2014.

[120] Xiaoyuan Suo. A study of graphical password for mobile devices. In *Mobile Computing, Applications, and Services. 5th International Conference, MobiCASE 2013, Paris, France, November 7-8, 2013, Revised Selected Papers*, pages 202–214, 2013.

[121] Y. Tang, Y. Shi, and X. Xiao. Influence maximization in near-linear time: A martingale approach. In *Proc. SIGMOD*, pages 1539–1554, 2015.

[122] Y. Tang, X. Xiao, and Y. Shi. Influence maximization: Near-optimal time complexity meets practical efficiency. In *Proc. SIGMOD*, pages 75–86, 2014.

[123] Pin Shen Teh, Andrew Beng Jin Teoh, and Shigang Yue. A survey of keystroke dynamics biometrics. *The Scientific World Journal*, 2013:408280–408280, 2013.

[124] D. Tennenhouse. Proactive computing. *Comm. ACM*, 43(5):43–50, 2000.

[125] D. Tennenhouse. Proactive computing: A progress report. MIT LCS Seminar, Dec 11 2002.

[126] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[127] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[128] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[129] Mike Wallace. *Jawbone Java WordNet API*, 2007.

[130] Mengting Wan and Julian McAuley. Modeling ambiguity, subjectivity, and diverging viewpoints in opinion question answering systems. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 489–498. IEEE, 2016.

[131] Y. Wang, G. Cong, G. Song, and K. Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In *Proc. SIGKDD*, pages 1039–1048, 2010.

[132] Yu Wang, Gao Cong, Guojie Song, and Kunqing Xie. Community-based greedy algorithm for mining top-k influential nodes in mobile social networks. In

*Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1039–1048. ACM, 2010.

[133] Ryen W White, Nicholas P Tatonetti, Nigam H Shah, Russ B Altman, and Eric Horvitz. Web-scale pharmacovigilance: listening to signals from the crowd. *Journal of the American Medical Informatics Association*, 20(3):404–408, 2013.

[134] Janyce M Wiebe, Rebecca F Bruce, and Thomas P O'Hara. Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 246–253. Association for Computational Linguistics, 1999.

[135] Hui Xu, Yangfan Zhou, and Michael R Lyu. Towards continuous and passive authentication via touch biometrics: An experimental study on smartphones. In *Symposium On Usable Privacy and Security, SOUPS*, volume 14, pages 187–198, 2014.

[136] Elad Yom-Tov and Evgeniy Gabrilovich. Postmarket drug surveillance without trial costs: discovery of adverse drug reactions through large-scale analysis of web search queries. *Journal of medical Internet research*, 15(6):e124, 2013.

[137] Yanyan Zhao, Bing Qin, Shen Hu, and Ting Liu. Generalizing syntactic structures for product attribute candidate extraction. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 377–380. Association for Computational Linguistics, 2010.

# Appendix A

# List of Publications

The following are the list of publications that have been produced during the course of this Ph.D. research.

**Published Work**

1. Chan, Philip K., and Ebad Ahmadzadeh. ”Improving efficiency of maximizing spread in the flow authority model for large sparse networks.” Big Data (Big Data), 2016 IEEE International Conference on. IEEE, 2016.

2. Crawford, Heather, and Ebad Ahmadzadeh. ”Authentication on the Go: Assessing the Effect of Movement on Mobile Device Keystroke Dynamics.” Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017). USENIX Association, 2017.

3. Ahmadzadeh, Ebad, and Philip K. Chan. ”Mining pros and cons of actions from social media for decision support.” Big Data (Big Data), 2017 IEEE International Conference on. IEEE, 2017.

**Work in Progress**

Ahmadzadeh, Ebad, and Philip K. Chan. ”Identifying Pros and Cons of Product Aspects Based on Customer Reviews”

Ahmadzadeh, Ebad, and Philip K. Chan. ”Forecasting of Sales Based on Buyer Intent Prediction, a Supervised Technique”