Introducing Semantic Role Labels and Enhancing Dependency Parsing to Compute Politeness in Natural Language

THESIS

Presented in Partial Fulfillment of the Requirements for the Degree Master of Science in the Graduate School of The Ohio State University

By

Smrite Dua, B.Tech.

Graduate Program in Computer Science and Engineering

The Ohio State University

2015

Master's Examination Committee:

Professor Eric Fosler-Lussier, Advisor

Professor Alan Ritter
Copyright by
Smrite Dua
2015
Abstract

Politeness is a crucial aspect of natural language and affects the way we communicate with others. Computing politeness in natural language can help analyze different social factors and derive relationships amongst them. Previous work to compute politeness in data primarily focuses on exploiting lexical information in the form of unigrams and specific terms and phrases (Danescu-Niculescu-Mizil et al., 2013). The main goal of this work is to analyze how semantic information can be combined with basic lexical information to compute politeness.

We build a politeness classifier that deploys basic lexical information using unigrams and combines it with semantic knowledge in the form of semantic role labels and dependency labels in our data. Experimental results for in-domain setup show improved performance of the politeness framework with the added semantic information and support our objective of evaluating politeness in a generic way by including language semantics than just certain lexical terms or phrases relating to politeness.

Additionally, we deploy classification rules in our model, as we believe rules can associate politeness more accurately and couple semantic and lexical insights in language at the same time. Our in-domain experiments verify the rationale behind including rules in the classifier model and support the hypothesis of politeness determination in a broad
way. Finally, we do more analysis on the top ranked semantic role label patterns with respect to their precision and coverage in the data.
Dedication

This document is dedicated to Nirmala Srivastava.
Acknowledgments

I was fortunate to work with the best minds at OSU. Firstly with all reverence, I would like to thank my advisor – Eric Fosler-Lussier for guiding me in my research. His immaculate advising style brings out the best in each of his students. I am very grateful to my committee member- Alan Ritter for his brilliant ideas on generic politeness rules and taking out time from his schedule to be on the examination committee.

Further, I want to express my gratitude to - Michael Mandel for always asking thought provoking questions and providing valuable suggestions. I am immensely thankful and lucky to have around smartest lab members – Andrew Plummer, Chaitanya Shivade, Dennis Griffis, Ryan He, Yi Ma, Joo-Kyung Kim, Deblin Bagchi, Sirui Xu and Young Suk.

Lastly, I would like to thank my family and all other folks for their love and support.
Vita

June, 2011 ......................................................B.Tech.(Major-Information Technology)
BVCOE, New Delhi

January, 2015 to present ...............................Graduate Teaching Associate, Department
of Computer Science and Engineering, The
Ohio State University

Fields of Study

Major Field: Computer Science and Engineering
Table of Contents

Abstract ............................................................................................................................... ii

Dedication .......................................................................................................................... iv

Acknowledgments ........................................................................................................... v

Vita ..................................................................................................................................... vi

Fields of Study ................................................................................................................ vi

List of Tables .................................................................................................................... x

List of Figures .................................................................................................................. xi

Chapter 1: Introduction ................................................................................................. 1

1.1 Organization of the Thesis ................................................................................... 2

Chapter 2: Preliminaries ............................................................................................... 4

2.1 Politeness Theory ................................................................................................. 4

2.1.1 Face ..................................................................................................................... 4

2.1.2 Face Threatening Acts ..................................................................................... 5

2.1.3 Politeness strategies ......................................................................................... 5

2.2 Semantic Role Labeling ....................................................................................... 7
Appendix A: Code Implementation of Feature Functions used in Previous Work ........ 43
List of Tables

Table 1: Key for different semantic roles ................................................................. 8

Table 2: Positive (1-5) and negative (6–20) politeness strategies used in Danescu-
Niculescu-Mizil et al. (2013) ................................................................................. 18

Table 3: Accuracies of their BOW and linguistic classifier for in-domain and cross-
domain settings. Human performance is included as a reference point. Table reproduced
from Danescu-Niculescu-Mizil et al. (2013). ......................................................... 20

Table 4: Percentage accuracies of the politeness construction model for Wikipedia in-
domain tests ........................................................................................................... 29

Table 5: Percentage accuracies of the politeness construction model for Stack Exchange
in-domain tests ......................................................................................................... 30

Table 6: Percentage accuracies of the politeness classifier in cross domain setting ...... 31

Table 7: Sample requests from Stack Exchange domain ....................................... 34

Table 8: Unigram weights from the Wikipedia and Stack Exchange domain .......... 35

Table 9: Impolite SRL patterns and their sample requests ..................................... 37

Table 10: Polite SRL patterns and their corresponding sample requests ............... 38
List of Figures

Figure 1: Stanford dependencies for a sentence ............................................................... 10
Figure 2: Basic dependencies representation................................................................. 10
Figure 3: Collapsed and Propagated dependencies representation............................. 11
Figure 4: Example requests in Stanford Politeness Corpus for Wikipedia domain......... 16
Figure 5: AM-MOD-A0-V-A1 patterns in a few Wikipedia’s polite requests ................. 23
Figure 6: A few JRip Rules learned for Wikipedia requests .......................................... 24
Figure 7: Semantic Role Label (SRL) parse of one of the request ................................ 25
Figure 8: A few N-gram SRL features ......................................................................... 26
Figure 9: Initial and final dependency parse for one of the request............................. 27
Figure 11: Scatter plot of the polite and impolite SRL patterns ................................... 36
Politeness in language affects the way we communicate with others and helps define different kinds of connections. Thus, an automated way of politeness determination in various types of text can decipher interesting social affinities and trends. Several engineering applications (or apps) can be built that deploy politeness computation. For instance an app designed to educate a physician on refining the politeness content of his language can manifest positive effects in a patient-physician relationship. A similar app can be designed for an academic setting where by training teachers will lead to better student-teacher bonding and overall academic performance. Further, a Facebook plug-in that would check the politeness content before a post is posted can help promote congenial relations on social media websites.

Danescu-Niculescu-Mizil et al. (2013) first proposed a model that estimates politeness using interpersonal requests from the Wikipedia and Stack Exchange online communities (section 3.1). Each request was classified as either polite or impolite, mainly based on the lexical information deployed in their model. On close analysis of the lexical information utilized in their model we find that their feature set consists of unigrams, sentiment words and several phrases and terms that relate to politeness in requests. Further, we find that
these lexical terms and phrases are specific to the data-type of politeness classification, in this case requests.

In this work, we explore the benefit of integrating semantic knowledge with basic lexical information to detect politeness in requests. The intuition behind utilizing semantic information is based on our observation of similar repeating semantic role label patterns in polite requests in the data set. Moreover, exploiting semantic information with basic lexical information forms a generic way to approach the problem of politeness determination and would be applicable to different data-types (initiating dialogues, giving warnings, giving advice, etc). First, we incorporate semantic knowledge in the form of semantic role labels and dependency labels, appending to a unigram baseline model, and examine our politeness framework for this setting. Then, we generate classification rules that combine the lexical and semantic insights in the data and include them as a feature of our classifier framework. We believe that rules render a definitive way of recognizing politeness and combine the effects of both semantic and lexical knowledge in the data at the same time. Thus, to existing baseline lexical features (unigrams) we add semantic role labels, dependency labels and rules as features to study politeness recognition for both domains (Wikipedia and Stack Exchange) and across these domains. Lastly, we do a deeper analysis on the highest ranked semantic role label structures in contrast with the precision and coverage of these structures in the data.

1.1 Organization of the Thesis

Chapter 2 outlines the relevant background knowledge required to understand the research ideas developed in this work. Chapter 3 describes the data set and previous work
on politeness computation as the baseline model we extend in our work. Finally, Chapter 4 describes new feature categories that we introduce, experimental settings, our results and analysis. To summarize, Chapter 5 reiterates the main contributions of our work and discusses possible future extensions.
Chapter 2: Preliminaries

This chapter will outline the necessary background material which forms the basis for understanding the ideas and concepts introduced in our work. We start with describing Politeness Theory, followed by Semantic Role Labeling, Dependency Parsing, Rule based classifiers and RIPPER classification method.

2.1 Politeness Theory

Politeness theory is broadly defined as a battery of social skills whose goal is to ensure everyone feels affirmed in a social interaction (Foley, 1997). This theory was first proposed by Penelope Brown and Stephen Levinson, 1978 and has greatly affected cognizance of politeness. In this section we describe different aspects of their politeness theory that we directly use in our work.

2.1.1 Face

Brown and Levinson’s notion of ‘face’ is based on that of Goffman (1967): a public self-image that is regularly involved in interaction and needs to be maintained. Face in terms of wants can be defined as:

- Positive face is the want of every individual that his interests be valuable to at least a few other interactants.
• Negative face is the want of every adult individual that his actions be unrestricted by other interactants.

2.1.2 Face Threatening Acts

Face threatening acts (FTA’s) are opposite in nature to the face needs of the addressee and/or of the speaker. In social interactions FTA’s are at times inevitable. Some of the examples of negative FTA’s are request, suggestion, warning or threat. Positive FTA’s examples include insults, complaints, disregard or disagreement.

2.1.3 Politeness strategies

These strategies are deployed to avoid FTA’s and mutual vulnerability of face that are inevitable in our day-to-day interactions. Here we discuss two main types of politeness strategies - positive and negative politeness strategies.

• Positive politeness strategies

As the name suggests, these strategies are directed towards hearer’s positive face by addressing his wants, including actions, acquisitions or values as desirable. Thus, positive politeness strategies are used to make the hearer feel good about himself, his interests or possessions and usually used in situations where the interactants know each other well (Foley, 1997). Some of the positive politeness strategies and their examples are listed as follows:

1. Attend to Hearer’s interests, needs, wants. For example- ‘What a beautiful vase this is! Where did it come from? ’

2. Usage of in-group identity markers – ‘Help me with this bag here, will you pal? ’
3. Being optimistic – ‘Look, I’m sure you won’t mind if I remind you to do the dishes tonight. ’

4. Include both speaker (S) and hearer (H) in activity – ‘Let’s stop for a bite. (i.e. I want a bite, so let’s stop) ’

5. Offer or promise – Even if these promises are false they form part of positive politeness strategies - ‘I’ll drop by sometime next week. ’

6. Exaggerate interest in hearer (H) and his interests – ‘What a lovely bracelet you have! Where did you buy it?’

7. Avoid Disagreement – ‘Yes, it’s permanent – permanent until I get married again. ’

8. Joke – ‘How about lending me this old heap of junk? ‘

9. Give (or ask) reasons - ‘Why not lend her your ring for the party?’

• Negative Politeness Strategies:

Negative politeness strategies are directed toward the addressee’s negative face and form the core of respect behavior. These strategies attend to the addressee’s need for freedom of unhindered action and unimpeded attention. Following we list some of the negative politeness strategies along with example sentences-

1. Be indirect – ‘Can you post this letter for me? ’

2. Use hedges or questions – ‘I guess that Harry is coming. ’

3. Be pessimistic – ‘Could you jump over the five-foot fence? ‘

4. Minimize the imposition – ‘I just want to ask you if I can borrow a single sheet of paper. ’
5. Use obviating structures, like nominalizations, passives, or statements of
general rules – ‘I’m sorry, but late-comers cannot be seated till the next
interval. ‘

6. Apologize – ‘I hope you don’t mind me saying this, but …’

7. Use plural pronouns – ‘We at Lockheed are not excessively concerned. ’

In the previous work chapter (Chapter 3), we will see how Danescu-Niculescu-Mizil et
al. (2013) relates these positive and negative politeness strategies to specific lexical
terms, and use these specific lexical terms in their classifier model to classify each
request as polite or impolite. Further, in this work we exploit semantic structure of
requests in the data set (section 3.1) to relate to these strategies. We also deploy rules that
use combination of semantic information (described in section 4.1) and unigrams in the
data to form associations with these strategies.

2.2 Semantic Role Labeling

Semantic role labeling (SRL), also called shallow semantic parsing, is a task of
identifying semantic arguments and their specific roles of the predicate or verb in a
sentence. A semantic representation is a higher-level of abstraction than a syntactic
representation (Gildea and Jurafsky, 2002). For instance ‘The car was sold by Allie to
Kevin’ has a different syntax tree, but the same semantic role labeling as ‘Allie sold the
car to Kevin’.

A semantic role in language is the relationship that a syntactic constituent has with a
predicate. Typical semantic arguments include Agent, Patient, Instrument, etc., as well as
adjunctive arguments indicating Locative, Temporal, Manner, Cause, etc. aspects. Table
1 lists key for different semantic roles that each constituent word can have with its predicate or verb in a sentence.

<table>
<thead>
<tr>
<th>Verb</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>V verb</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Arguments</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A0 subject</td>
<td></td>
</tr>
<tr>
<td>A1 object</td>
<td></td>
</tr>
<tr>
<td>A2 indirect object</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C-arg continuity of an argument/adjunct of type arg</td>
<td></td>
</tr>
<tr>
<td>R-arg reference to an actual argument/adjunct of type arg</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adjunct</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>AM-ADV adverbial modification</td>
<td></td>
</tr>
<tr>
<td>AM-DIR direction</td>
<td></td>
</tr>
<tr>
<td>AM-DIS discourse marker</td>
<td></td>
</tr>
<tr>
<td>AM-EXT extent</td>
<td></td>
</tr>
<tr>
<td>AM-LOC location</td>
<td></td>
</tr>
<tr>
<td>AM-MNR manner</td>
<td></td>
</tr>
<tr>
<td>AM-MOD general modification</td>
<td></td>
</tr>
<tr>
<td>AM-NEG negation</td>
<td></td>
</tr>
<tr>
<td>AM-PNC proper noun component</td>
<td></td>
</tr>
<tr>
<td>AM-PRD secondary predicate</td>
<td></td>
</tr>
<tr>
<td>AM-PRP purpose</td>
<td></td>
</tr>
<tr>
<td>AM-REC reciprocal</td>
<td></td>
</tr>
<tr>
<td>AM-TMP temporal</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Key for different semantic roles
(http://cogcomp.cs.illinois.edu/page/demo_view/srl)
Semantic representation in form of SRL helps to answer ‘Who’, ‘When’, ‘What’, ‘Where’ and ‘Why’ questions which have direct application in Information Extraction, Question Answering, Summarization and other NLP tasks.

Following sentence exemplifies the annotation of semantic roles:

\[
[A_0 \text{They}][V \text{had}] [A_1 \text{brandy}] [AM-LOC \text{in the library}]
\]

For this work, we use semantic role labeling to explore how politeness in language connects to its semantic representation. Moreover, SRL parses of polite requests suggest similar repeating SRL patterns, which are also produced as a part of a classification rule in section 4.1.

2.3 Stanford typed Dependencies

Stanford typed dependencies (de Marneffe and Manning, 2006) render a clear description of the grammatical relationships in a sentence that is easy to use and understand by individuals without linguistic expertise who want to extract textual relations. Stanford dependencies (SD) are triplets: name of the relation, governor and dependent. Figure 1 shows standard dependencies for the sentence – ‘Bills on ports and immigration were submitted by Senator Brownback, Republican of Kansas’. Figure 2 depicts basic dependencies representation and figure 3 shows graphical representation of Stanford dependencies called as collapsed and propagated representation.
nsubjpass(submitted, Bills)
auxpass(submitted, were)
agent(submitted, Brownback)
nn(Brownback, Senator)
appos(Brownback, Republican)
prep_of(Republican, Kansas)
prep_on(Bills, ports)
conj_and(ports, immigration)
prep_on(Bills, immigration)

Figure 1: Stanford dependencies for sentence—‘Bills on ports and immigration were Submitted by Senator Brownback, Republican of Kansas.’ Reproduced from http://nlp.stanford.edu/software/stanford-dependencies.shtml

Figure 2: Basic dependencies representation, reproduced from http://nlp.stanford.edu/software/stanford-dependencies.shtml
Figure 3: Collapsed and Propagated dependencies representation, reproduced from-

From basic dependencies graph (Figure 2) it is apparent that they form a directed acyclic
graph. In this type of representation each word in a sentence is dependent of exactly one
inghing, either another word or ‘ROOT’ word. In collapsed representation, dependencies
involving prepositions and conjuncts are collapsed to get direct dependencies between
content words. Also in case of conjunction, you can get propagation of the dependencies
involving conjuncts which produces collapsed and propagated representation. In figure 3
conjunction between ‘ports and immigration’ leads to propagation of ‘prep_on’ relation
from first conjunct-‘ports’ to second conjunct-‘immigration’ as well.

In this work, we use basic dependency representation for each request (in the data) as we
believe that each individual grammatical relation of a word would contribute to any
structural pattern that could relate to politeness and collapsing (combining) these dependency relations might constrain these semantic patterns for our analysis.

2.4 Rule based classifiers and RIPPER classification method

In this section, we first consider rule based classifiers, and then describe RIPPER classification which we deploy to generate politeness rules in our model.

2.4.1 Rule based classifiers

In this kind of classifiers each classification rule has the following construct:

\[ r: (\text{Condition}) \rightarrow y, \]

where the left hand side condition (LHS) is the conjunction of attributes and right hand side \(y\) (RHS) is the class label. A rule \(r\) is said to cover an example in the data if the attributes of the example meet the condition of the rule.

Rule based classifiers implement few techniques to enforce some of the rule’s properties and higher classification accuracy:

1. Techniques to ensure rules are mutually exclusive: To deploy mutual exclusiveness of rules it is required that new rules are generated from new examples that avoid overlapping coverage. In addition, rules follow priority based ordering. Voting can be used to allow an example to form multiple rules, and render a voting scheme for each consequent class of the rule.

2. Techniques to ensure that rules are exhaustive: Classifier rules can be made exhaustive by adding a default rule – \( r': (\) \rightarrow y' \)
Where default rule (represented as $r'$) has an empty condition – ( ) and is picked up when none of the generated rule is satisfied. Rule consequent - $y'$ is the default class or the majority class.

### 2.4.2 RIPPER classification method:

RIPPER classification method (Cohen, 1995) broadly consists of three phases – growing a rule, building a rule set and optimizing the rule set.

1. **Growing a rule:** A rule starts as an empty conjunction of conditions and conjuncts are added to it till they improve FOIL’s information gain \(^1\) and the rule no longer covers a negative example. Then incremental reduced error pruning is applied based on metric (equation 1) which guides pruning in an intuitive way and selects rules that are more predictive.

   \[
   v = \frac{p - n}{p + n} \quad \text{(Equation 1)}
   \]

   So the conditions that maximize $v$ metric are pruned from the rule. Here $p$ is the number of positive examples covered by the rule in the validation set and $n$ is the number of negative examples covered by the rule in validation set.

2. **Building a rule set:** This phase of RIPPER algorithm uses sequential covering algorithm, where we find the best rule for the current set of positive examples. Then we add this rule to the rule set and compute the description length (description length is the number of bits needed to code current rule set, and their examples) and we stop adding rules to the rule set when the new computed description length exceeds the shortest computed description length by a fixed

---

\(^1\) FOIL information gain($c;r$) = $t \times (\log_2 (p1/p1 + n1) - \log_2(p0/p0 + n0))$, Where $c$ is the conjunct added to rule $r$, and $t$ is the number of positive examples covered by both $r$ and $c$ with $r$. 

13
defined limit. After adding the rule to the rule set we eliminate its positive and negative examples covered in the dataset and repeat the same procedure with a new rule.

3. Rule Optimization: In this stage each rule is processed in the order it appears in the rule set. For each rule two new rules are created – a revision rule and a replacement rule. A revision rule is obtained by adding conjuncts to the original rule, whereas a replacement rule is formed from scratch for the original rule and minimizes the pruning error on the rule set. Then we compare three rule sets - original rule set, rule set with revision rule in place of the original rule and third rule set with the replacement rule in place of the original rule and chose the rule set that minimizes description length.

Ripper rule based classification is available in Weka (Hall et al., 2009) as ‘JRip’ classification method that we utilize in our work to obtain politeness rules. These class rules help combine lexical and semantic information in the data and exhibit highest priority over all other features used in our model (refer section 4.4).
Chapter 3: Previous work

In this chapter, we describe the implementation of politeness classifier of Danescu-Niculescu-Mizil et al. (2013), whose model we extend in our work.

3.1 Data set

In the 2013 study, a corpus of requests was collected, called the Stanford Politeness Corpus. Each request in the corpus carries a binary politeness label (i.e. polite or impolite). A request is selected as a data-point for the task of politeness classification when it involves an imposition on the addressee, and thus would allow investigation of socio-linguistic facets of politeness. Each request is made up of two sentences, where the second sentence forms the actual request and the first sentence gives the context of the request. Figure 4 enumerates some of the requests in Stanford Politeness Corpus.
<table>
<thead>
<tr>
<th>Request</th>
<th>Politeness Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hi David, you're good with images. Anything you can do to clean up &lt;url&gt; so it's useful on ITN?</td>
<td>P</td>
</tr>
<tr>
<td>Hi, I understand that the WP:COI rules would still apply, regardless of username. Could you kindly approve the change in my user name from cecn to funshine?</td>
<td>P</td>
</tr>
<tr>
<td>(Constantine and Leo are English, not Latin names.) Why don't you answer my question about Constantine the Great?</td>
<td>I</td>
</tr>
</tbody>
</table>

Figure 4: Example requests in Stanford Politeness Corpus for Wikipedia domain

Furthermore, according to Danescu-Niculescu-Mizil et al. (2013), these requests solicit specific information or action and expect a response and thus can be used to explore how politeness relates to different social factors.

Requests are extracted from two online communities – the Wikipedia community of editors and the Stack Exchange question-answer community. There are 35,661 requests from Wikipedia, out of which 4,353 are annotated, and 373,519 requests from Stack Exchange, out of which 6,603 are annotated. Wikipedia requests are centered around the creation and maintenance of the collaborative encyclopedia, whereas Stack Exchange requests are comments from users on posts asking for more information or proposing edits.

The Stanford Politeness Corpus contains following Metadata:

- Timestamps – (From Wikipedia and Stack Exchange)
- Editor’s Status – (From Wikipedia)
• User’s Reputation (From Stack Exchange): A rough measurement of how much the community trusts given user and is earned by convincing peers about user’s knowledge in particular field or area.

• Up/Down votes – (From Stack Exchange)

3.2 Politeness Strategies and Feature Functions

Table 2 lists different positive and negative politeness strategies used and their corresponding markers in requests in the dataset. Feature functions are divided into three categories depending on the form in which request was given as input –

• Dependency-based Feature Functions – These features used the dependency parse of a request from the Stanford Dependency Parser (de Marneffe et al., 2006) as input to determine the presence or absence of the given politeness strategy in the request. For example, for the ‘Direct Question’ politeness strategy, its dependency-based feature function would check if governor (left word) or dependent (right word) is ‘Why’, ‘What’, ‘Who’ or ‘How’ and occurs at the start of sentence in the request. Similarly, for a ‘Hedges’ politeness strategy, its feature function would check if the dependency parse of the request contains a ‘nsubj’ dependency relation with its left word belonging to one of the words in the hedge word list (Hyland 2005).

• String-based Feature Functions – Here the text form of request is given as input to determine the presence or absence of the given politeness strategy. For instance, for the ‘Counterfactual Modal’ politeness strategy, its string-based feature
function would check for the presence of ‘could you’ or ‘would you’ string in the text input.

- **Token-based Feature Function** – As the name suggests, input to this feature function is in the form of tokens of request. For example, for the ‘Positive Lexicon’ politeness strategy, its token-based feature function would check for any tokens in the request that belong to the positive lexicon list (Liu et al., 2005).

Implementation of these feature functions for different politeness strategies is given in Appendix A.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Gratitude</td>
<td>I really <strong>appreciate</strong> that you’ve done them.</td>
</tr>
<tr>
<td>2 Deference</td>
<td><strong>Nice work</strong> so far on your rewrite.</td>
</tr>
<tr>
<td>3 Greeting</td>
<td><strong>Hey,</strong> I just tried to…</td>
</tr>
<tr>
<td>4 Positive lexicon</td>
<td><strong>Wow!</strong> This is a great way…</td>
</tr>
<tr>
<td>5 Negative lexicon</td>
<td>If you’re going to <strong>accuse me…</strong></td>
</tr>
<tr>
<td>6 Apologizing</td>
<td><strong>Sorry</strong> to bother you…</td>
</tr>
<tr>
<td>7 Please</td>
<td>Could you <strong>please</strong> say more…</td>
</tr>
<tr>
<td>8 Please start</td>
<td><strong>Please</strong> do not remove warnings…</td>
</tr>
<tr>
<td>9 Indirect(btw)</td>
<td><strong>By the way,</strong> where did you find…</td>
</tr>
<tr>
<td>10 Direct question</td>
<td><strong>What</strong> is your native language?</td>
</tr>
<tr>
<td>11 Direct start</td>
<td><strong>So</strong> can you retrieve it or not?</td>
</tr>
<tr>
<td>12 Counterfactual modal</td>
<td><strong>Could/would</strong> you…</td>
</tr>
<tr>
<td>13 Indicative modal</td>
<td><strong>Can/will</strong> you…</td>
</tr>
<tr>
<td>14 1(^{st}) person start</td>
<td>I have just put the article…</td>
</tr>
<tr>
<td>15 1(^{st}) person pl.</td>
<td><strong>Could we</strong> find a less complex name…</td>
</tr>
<tr>
<td>16 1(^{st}) person</td>
<td>It is <strong>my</strong> view that…</td>
</tr>
<tr>
<td>17 2(^{nd}) person</td>
<td>But what’s the good source <strong>you</strong> have in mind?</td>
</tr>
<tr>
<td>18 2(^{nd}) person start</td>
<td><strong>You’ve</strong> reverted yourself</td>
</tr>
<tr>
<td>19 Hedges</td>
<td>I suggest we start with…</td>
</tr>
<tr>
<td>20 Factuality</td>
<td><strong>In fact</strong> you did link…</td>
</tr>
</tbody>
</table>

Table 2: Positive (1-5) and negative (6–20) politeness strategies used in their politeness framework. Positive politeness strategies address the desire to be paid respect, while negative politeness strategies address the desire not to be told what to do. Table reproduced from Danescu-Niculescu-Mizil et al. (2013)
3.3 Experiments

In this section, we discuss the experimental set-up of the politeness framework by Danescu-Niculescu-Mizil et al. (2013), and results and analysis of their work. For annotated data in Stanford Politeness corpus (described in section 3.3) both polite and impolite class are balanced with each class having 1,089 requests for the Wikipedia domain and 1,651 requests for the Stack Exchange domain.

For the baseline model, an SVM classifier is utilized using unigrams as features. Unigrams appearing less than ten times are not included in the model. This baseline of unigrams (or bag of words, BOW model) is extended to include different linguistic features pointed out in section 3.2. To obtain a reference point for their models and to compare the performance of the BOW model and the linguistically informed model, they collect three new politeness annotations for their data and then calculate human performance for this classification task by the percentage match between the average scores of the three new annotations to the original annotation of the data. So a positive average score would match a class label of ‘polite’ in the original annotation of the data.

Table 3 lists the comparison of these three models for both the in-domain and cross-domain settings. For the in-domain setting, leave-one-out cross validation is employed, while for the cross-domain setting they train on one domain and test on another. From Table 3, the linguistically informed model of in-domain and cross-domain gives 3-4% absolute improvement over the BOW model. Also, Danescu-Niculescu-Mizil et al. (2013) claim that their theory inspired features are indeed effective and generalize well to other domains.
<table>
<thead>
<tr>
<th>Train Test</th>
<th>Wiki</th>
<th>SE</th>
<th>Wiki</th>
<th>SE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>79.84%</td>
<td>74.47%</td>
<td>64.23%</td>
<td>72.17%</td>
</tr>
<tr>
<td>Ling.</td>
<td>83.79%</td>
<td>78.19%</td>
<td>67.53%</td>
<td>75.43%</td>
</tr>
<tr>
<td>Human</td>
<td>86.72%</td>
<td>80.89%</td>
<td>80.89%</td>
<td>86.72%</td>
</tr>
</tbody>
</table>

Table 3: Accuracies of their BOW and linguistic classifier for Wikipedia (Wiki) and Stack Exchange (SE), for in-domain and cross-domain settings. Human performance is included as a reference point. Table reproduced from Danescu-Niculescu-Mizil et al. (2013).

3.4 Summary

This chapter discussed Stanford politeness corpus and the experimental settings used by Danescu-Niculescu-Mizil et al. (2013) for politeness computation in requests. We deploy the same data set for our implementation of politeness framework in the next chapter. We observe that feature functions used in this work utilizes only lexical knowledge in the data using unigrams, sentiment words or certain terms or phrases. In the next chapter we extend baseline model of this work to include semantic information in the data.
Chapter 4: Experiments

This chapter describes our implementation of the politeness framework (Danescu-Niculescu-Mizil et al., 2013) and enhance the previous model to include new categories of features to analyze how these new features influence politeness computation in the data set (described in section 3.1).

4.1 Feature Categories

As pointed out previously, each request in the dataset is classified as polite or impolite by the politeness framework. Here we explain our implementation of baseline features of Danescu-Niculescu-Mizil et al. (2013) (described in section 3.3) and different categories of new features we use to extend their model. For each feature type, we also point out the rationale behind their use in our model.

- **Unigrams** - We use unigrams that appear more than ten times in the request dataset; this is the same as the baseline feature used in Danescu-Niculescu-Mizil et al. (2013) described in section 3.3.

- **Token count** - Total number of tokens in each request forms the token count feature, which is a new feature that we consider to extend the baseline model. We consider the token count feature to be a measure of reasoning in each request. Reasoning is listed as one of the positive politeness strategies in politeness theory.
(by Brown & Levinson, 1987; refer section 2.1) and we use the token the count feature to examine how reasoning can help our politeness classifier decode politeness in the request dataset.

- Semantic Role Labels (SRL) - As described in section 2.2, semantic role labeling is the task of identifying semantic arguments associated with the verb or predicate of the sentence. We apply N-gram modeling to the semantic role label parse of each request in our data to inspect how the semantic structure of a request relates to its politeness. Thus, we employ N-gram semantic role label as a new feature in our model.

The rationale behind using the N-gram semantic role label feature is based on our observations of the dataset: for several polite requests in the data we observed similar repeating N-gram semantic role label patterns involving the AM-MOD argument. Figure 5 highlights repeating AM-MOD-A0-V-A1 semantic role label pattern in Wikipedia’s polite requests.
Regarding our WikiProject discussion about airport naming I am unsure about Wiki etiquette - should I edit your sandbox with links indicating the common name (i.e. airport even though it's an aerodrome) or should I list them on my own talk page?

I appreciate your effort to seek consensus through a merge proposal before merging <url> to <url> but as you are no doubt aware no one participated in the discussion meaning that while there was no articulated consensus against it there was also no consensus for it. May I recommend undoing the merge for now re-opening the discussion and seeking wider input at the relatively large number of WikiProjects that handle those articles?

I was in doubt too but when dealing with such incomprehensible concoctions I prefer to err on the side of BLP caution. Would you like it userfied?

I tried to create a similar map for Canada but couldn't get a usable output from the site you used for the Australia map. Could you create a Canadian one?

Figure 5: AM-MOD-A0-V-A1 patterns in a few Wikipedia’s polite requests

Furthermore, the N-gram SRL feature – AM-MOD-A0-V-A1 is selected among the top three highest ranked features in our classifier model involving unigrams and SRL features.

- Dependency Labels (DL) - Dependency labels, as discussed in section 2.3, define the grammatical relationship of words (syntactic units) in a sentence to the verb (structural center for a given sentence). We apply a similar N-gram model to the dependency parse of each request and then use these N-grams of dependency labels as features in our model.

In contrast to the dependency feature functions used in previous work (which finds a particular grammatical relation and its corresponding governor/dependent word, Section 3.2 and Appendix A), we utilize the N-gram model of dependency...
labels as our new feature for similar reasons as using the N-gram SRL feature. Hence, by including dependency labels we intend to relate the structure and the grammatical relations in a sentence to its politeness content.

- Rules: We use the JRip algorithm (discussed in section 2.4.2) to obtain class rules, which form a new category of features in our implementation model. Each class rule consists of the conjunction of different feature types on the left hand side and a politeness label on the right hand side. These rules allow us to combine different feature categories together as a single feature in our politeness classifier and then use this new feature to evaluate politeness in each request. These rules allow us to inspect the effect of combining baseline features (unigrams) with new categories of features (SRL, DL or token length) in our implementation model. Moreover, combining different types of features as rules gives a more definitive way of defining politeness in the data. Figure 6 lists the rules obtained for Wikipedia domain.

<table>
<thead>
<tr>
<th>(SRL-AM-MOD A0 V A1 &gt;= 1) and (Unigram-I &gt;= 1) =&gt; Class = Polite</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Unigram-I &gt;= 1) and (Unigram-why &lt;= 0) and (Unigram-you &gt;= 1) and (Unigram-&lt;url&gt;= 1) =&gt; Class = Polite</td>
</tr>
<tr>
<td>(SRL-AM-MOD A0 V A1 &gt;= 1) and (Unigram-please &gt;= 1) =&gt; Class = Polite</td>
</tr>
<tr>
<td>(Unigram-I &gt;= 1) and (Unigram-why &lt;= 0) and (SRL-A1 V AM-ADV A0 &gt;= 1) =&gt; Class = Polite</td>
</tr>
</tbody>
</table>

Figure 6: A few JRip Rules learned for Wikipedia requests
4.2 Preprocessing Data

Here we outline the different kinds of preprocessing required of the request data set in order to obtain baseline features (unigrams) and the new categories of features for our employment of the politeness framework.

- **Preprocessing for Unigram Features** – To obtain Unigrams occurring more than ten times in the request data set, we use Weka’s ‘StringToWordVector’ filter. Thus after applying Weka’s ‘StringToWordVector’ filter with an appropriate configuration, we obtain unigrams feature for our model.

- **Preprocessing for Semantic Role Labels Features** – Each request in the data is passed through a semantic role labeler. The output of the semantic role labeler is a semantic role label parse of the given request. Figure 7 shows a semantic role label parse for one of the requests. Now we apply N-gram modeling to the semantic role label parse to get N-gram semantic role labels as features.

<table>
<thead>
<tr>
<th>Request</th>
<th>SRL parse for first sentence</th>
<th>SRL parse for second sentence</th>
</tr>
</thead>
<tbody>
<tr>
<td>I fail to see how you call copyleft ''selfish'' but are fine with the idea of totally proprietary licenses. Wouldn't those be even more selfish then?</td>
<td>(A1*) (V*) (A2* *********<em><em><em>) *** ************ (A0</em>) ** (V</em>) (A1</em> *** *******<em><em><em><em><em>) ************ (R-AM-MNR</em>) (A0</em>) (V</em>) (A1</em>) (A2</em> <em><em>) ************ (V</em>) (AM-MNR</em>) ** ********</td>
<td>(A1* <em><em>) (V</em>) (A2</em> <em>) (AM-TMP</em>) *</td>
</tr>
</tbody>
</table>

Figure 7: Semantic Role Label (SRL) parse of one of the request
We use SENNA (Collobert et al., 2011) software to obtain the semantic role label parse of each input request and then apply the ‘StringToWordVector’ filter in Weka to the semantic role label parse output. Figure 8 shows a few of the N-gram semantic role label features used in our model.

@attribute 'SRL-V A1 AM-MNR A0'
@attribute 'SRL-V A1 AM-TMP A0 V'
@attribute 'SRL-A1 AM-MOD A0 V A1'
@attribute 'SRL-V AM-ADV A0 V'

Figure 8: Few N-gram SRL features

- Preprocessing for Dependency Labels Features - First we input each request to a dependency parser, which produces the dependency parse of the given request. Then we apply similar N-gram modeling to the dependency parse produced (as we do to the SRL parse) to get the N-gram dependency labels features. We use Stanford Dependency Parser (de Marneffe and Manning, 2008) to produce the dependency parse of each request. Basic dependencies (as described in section 2.3) are used to represent the dependency parse for the request. For each word in the sentence and its Stanford Dependency triplet, we only keep the dependency relation and discard the governor and dependent words of the triplet. We also append the root verb to the start of the sentence. Figure 9 depicts the initial and final dependency parse of a given request. Finally, we apply the
‘StringToWordVector’ filter in Weka to the final dependency parse to obtain N-gram dependency labels features.

<table>
<thead>
<tr>
<th>Request</th>
<th>Initial Dependency Parse</th>
<th>Final Dependency parse</th>
</tr>
</thead>
<tbody>
<tr>
<td>I see that you closed that CfD discussion as rename but haven't actually renamed the categories. What's up with that?</td>
<td>First Sentence: nssubj(see-2, I-1) root(ROOT-0, see-2) mark(closed-5, that-3) nsbj(closed-5, you-4) ccomp(see-2, closed-5) det(discussion-8, that-6) nn(discussion-8, CfD-7) dobj(closed-5, discussion-8) prep(closed-5, as-9) pobj(as-9, rename-10) cc(closed-5, but-11) aux(renamed-15, have-12) neg(renamed-15, n't-13) advmod(renamed-15, actually-14) conj(closed-5, renamed-15) det(categories-17, the-16) dobj(renamed-15, categories-17)</td>
<td>First sentence: see nsbj mark nsubj ccomp det nn dobj prep pobj cc aux neg advmod conj det dobj Second sentence: 's nsubj advmod prep pobj</td>
</tr>
</tbody>
</table>

Figure 9: Initial and final dependency parse for one of the request

- Preprocessing for Token Count Features–The total number of tokens in the given request forms the token count feature and is obtained using the SENNA software,
which outputs the token count of the input request. Further, we apply Weka’s ‘discretize’ filter to get bins for numeric values of the token count feature.

- Preprocessing for Rules Features– As mentioned previously, we employ classification rules in our model, where we use several combinations of unigrams, N-gram SRL’s, N-gram DL’s and token count features. To generate these rules, we use Weka’s ‘JRip’ classification algorithm and then we employ these rules as features in our politeness classifier.

4.3 Task Set-up

In this section we report the experimental set-up for our politeness classifier and different experiments that we perform to gain insight into the request data set. We use Weka’s ‘Simple Logistic’ classifier for classifying each request as polite or impolite with various feature combinations. First, we experiment with the baseline features (unigrams) combined individually with the new feature categories to determine the effect each new category feature has on the baseline model. In addition, we test the combination of new features with the baseline model to determine if adding new features together also has an additive effect on the performance of our classifier. To test the robustness of our new features, we perform similar cross-domain tests (as in Table 3) between Wikipedia and Stack Exchange domains. Tables 4 and 5 list the performance of our politeness framework with different features for in-domain setting in Wikipedia and Stack Exchange respectively and Table 6 lists the performance of the classifier in cross-domain setting. Table 3 from the previous work chapter is juxtaposed for comparison.
<table>
<thead>
<tr>
<th>Train Test</th>
<th>Wiki Wiki</th>
<th>SE SE</th>
<th>Wiki SE</th>
<th>SE Wiki</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOW</td>
<td>79.84%</td>
<td>74.47%</td>
<td>64.23%</td>
<td>72.17%</td>
</tr>
<tr>
<td>Ling.</td>
<td>83.79%</td>
<td>78.19%</td>
<td>67.53%</td>
<td>75.43%</td>
</tr>
<tr>
<td>Human</td>
<td>86.72%</td>
<td>80.89%</td>
<td>80.89%</td>
<td>86.72%</td>
</tr>
</tbody>
</table>

Table 3: Accuracies of BOW and linguistic classifier for Wikipedia (Wiki) and Stack Exchange (SE), for in-domain and cross-domain settings. Human performance is included as a reference point. Reproduced from Danescu-Niculescu-Mizil et al. (2013).

<table>
<thead>
<tr>
<th>Unigrams</th>
<th>SRL</th>
<th>DL</th>
<th>Token Count</th>
<th>Rules</th>
<th>Percentage Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>79.68</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>80.97</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>80.69</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>80.28</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>80.74</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>81.38*</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>80.87</td>
</tr>
</tbody>
</table>

Table 4: Percentage accuracies of the politeness construction model for Wikipedia in-domain tests
<table>
<thead>
<tr>
<th>Unigrams</th>
<th>SRL</th>
<th>DL</th>
<th>Token Count</th>
<th>Rules</th>
<th>Percentage Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>62.09</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>62.15</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td>✓</td>
<td></td>
<td></td>
<td>62.21</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>61.15</td>
</tr>
<tr>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>62.67</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>62.27</td>
</tr>
</tbody>
</table>

Table 5: Percentage accuracies of the politeness construction model for Stack Exchange in-domain tests
<table>
<thead>
<tr>
<th>Unigrams</th>
<th>SRL</th>
<th>DL</th>
<th>Token Count</th>
<th>Rules</th>
<th>Percentage Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>61.12</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>60.45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>60.54</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>61.39</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>60.96</td>
<td></td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>59.57</td>
</tr>
</tbody>
</table>

| ✓ | ✓ | ✓ | ✓ | ✓ | 69.79 |
| ✓ | ✓ | ✓ | ✓ | ✓ | 69.10 |

Table 6: Percentage accuracies of the politeness classifier in cross domain setting

4.4 Analysis

In this analysis section, we discuss both the in-domain (Tables 4 and Table 5) and cross domain (Table 6) settings of the politeness classifier. For in-domain experiments, we train our model on one domain (Wikipedia or Stack Exchange) and then test it on the
same domain. However, for cross-domain experiments we train on one domain and test on the other.

For the Wikipedia domain, each add-one-in experiment of our new features with the baseline model raises the performance of the classifier. From Table 4 we see that adding N-gram SRL features to the unigrams (baseline) gives a gain of 1.3% in the classifier performance. This result supports our observations of the data set (as mentioned previously in this chapter) and our intuition for using N-gram SRL features, and thus suggests that semantic structure in language can be exploited to decode politeness. For the add-one-in experiment with N-gram DL features, we observe a rise around 1% which further supports our rationale behind using DL’s (along similar lines as the N-gram SRL features), and suggests that politeness in language can be related to patterns of semantic information in it. We also observe an improvement of 0.6% for including the token count feature with unigrams in our model. This increase suggests that reasoning can be measured with respect to the total number of tokens in a sentence and thus can indirectly account for politeness in the data, as reasoning is listed as one of the positive politeness strategies (Brown & Levinson, 1987). Rules appended to the baseline model, provide an increase of 1.1% in accuracy. We believe that class rules using a combination of unigrams with new features (N-gram SRL’s, N-gram DL’s and token count) define politeness in the data in a more concrete fashion. However, as the gain from rules is not significant when compared to the gain from the other add-one-in tests, we infer that more investigation is needed to formulate rules that will define politeness more accurately in the data.
We get the highest gain of 1.7% for the classifier performance when we use unigrams with all of the new features. This result is statistically significant with p-value equal to 0.029 for McNemar’s test (Deitterich, 1998). A gain of this type suggests that each new feature contributes to the increase in the classifier’s performance in its own way and these improvements produce an additive effect when we deploy all new features together with the unigram model.

From Table 5 of Stack Exchange in-domain tests, we observe that the add-one-in experiments of either SRL or DL or token count features alone do not boost the performance of our classifier. And the add-one in experiment for rules with the baseline model gives a slight improvement of 0.5% to the classifier performance. Request data from the Stack Exchange consists of requests that are comments to the posts of users discussing different topics such as Programming, Cooking, Photography, etc. These comments revolve around information clarification, suggesting corrections to the post or seeking any meta-information for answering the post question. Table 7 enumerates different kinds of annotated Stack Exchange requests, where the first two requests ask the user for editing his post question, third and fifth requests demand meta information for answering the original post question, and request number fourth and sixth seek to clarify the post question, for it to be answered correctly. Therefore, we believe that these varying styles of comments as requests are differently structured and lead to poor classifier performance for our new category features which look for certain semantic structures in these requests.

---
2http://meta.stackexchange.com/questions/19756/how-do-comments-work
Table 7: Sample requests from Stack Exchange domain

<table>
<thead>
<tr>
<th>#</th>
<th>Request</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I'm sorry, but I'm really struggling to understand what the problem is. Would you mind rephrasing the text after the first code sample?</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td>You should change the title to a question. Maybe something like &quot;What is the most popular Dictionary Application for Ubuntu?&quot;</td>
<td>I</td>
</tr>
<tr>
<td>3</td>
<td>How are you getting your data? ArcGIS Server feeds? SDE? St code sample?</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td>Title say permutation, but body says combinations. Which is it planning about?</td>
<td>I</td>
</tr>
<tr>
<td>5</td>
<td>Good question! What formula did you attempt, and in what situations was it wrong?</td>
<td>P</td>
</tr>
<tr>
<td>6</td>
<td>What do you mean by &quot;how to charge&quot;? Are you asking how much money to charge or how to implement an ad service and payment gateway?</td>
<td>I</td>
</tr>
</tbody>
</table>

Table 6 shows cross domain results where features from Wikipedia domain are applied to Stack Exchange and vice versa to see how well our new feature categories from one domain do for the other domain. Results obtained under this scenario are disappointing as we observe a drop in our classifier performance for any combination of the new features with the baseline model. Firstly, this poor cross-domain performance can be attributed to the composition of a request in the Stack Exchange community (Table 7) and similar justification we give for the poor in-domain Stack Exchange results (Table 5). Secondly, editors on Wikipedia are expected to follow a defined set of behavioral guidelines. These guidelines are rooted around cooperation and collaboration among fellow Wikipedians to build an international online encyclopedia. Henceforth, requests from the Wikipedia domain are more probable to be polite and homogeneously-structured as:

opposed to the requests from the Stack Exchange which have divergent structures.

Thirdly, our in-domain baseline model of the Wikipedia domain has higher weights assigned (Table 8) to unigrams such as ‘Thanks’, ‘Please’, ‘Hi’, ‘Help’ than the Stack Exchange domain, which supports our reasoning behind the differences in construction of the requests from two different domains, and hence, a low performance of our framework when applied across domains.

<table>
<thead>
<tr>
<th>Unigrams</th>
<th>Wikipedia In-domain weights</th>
<th>SE In-domain weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Please</td>
<td>.034</td>
<td>.012</td>
</tr>
<tr>
<td>Thanks</td>
<td>.055</td>
<td>.011</td>
</tr>
<tr>
<td>Hi</td>
<td>.046</td>
<td>.002</td>
</tr>
<tr>
<td>Help</td>
<td>.042</td>
<td>.003</td>
</tr>
</tbody>
</table>

Table 8: Unigram weights from the Wikipedia and Stack Exchange domain

4.5 More Analysis on SRL patterns

In this section we investigate the top ranked SRL patterns obtained using unigrams and rules derived via the RIPPER algorithm in our classifier model, and their correspondence with respect to polite and impolite request instances.
In Figure 11, we plot precision of each polite and impolite SRL pattern (top ranked) against the total number of instances covered by the pattern. We see a negative non-linear correlation between the polite SRL patterns as opposed to the total number of instances covered by them, which indicates that the SRL patterns that relate to politeness in data more effectively (i.e. with high precision) are usually less prevalent in the data. We observe a similar trend for the impolite SRL patterns that depict that high precision impolite SRL patterns are less likely to be seen in the data. However, the outlier for the polite SRL pattern which maps to AM-MOD A0 V A1, exhibits a widespread presence in the data as it lies among the top three features selected in our politeness classifier involving unigrams and SRL features.

\*Precision of a single rule: when it fires, what percentage of the time does it accurately predict the polite (impolite) class if this is the only rule in the classifier.\*
Table 9 and Table 10 show the impolite and polite SRL patterns, their mapping to example requests, and their precision and total count values. In Table 9, we notice how the SRL pattern – ‘A0 AM-NEG V A1’ maps to phrases – ‘you not read his request’ and ‘you not insist that the starting sample space’ in sample requests. Investigating these SRL structures in contrast with the number and the type of instances covered, and their precisions further validates our intuition of existence of specific semantic structures in the request data that directly encode politeness in the data.

<table>
<thead>
<tr>
<th>SRL Pattern</th>
<th>Sample Requests</th>
<th>Precision, # of Requests containing the Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>A0 AM-ADV V A1</td>
<td>Back on topic Jigglyfidders; I totally agree with Deconstructthis on this one. What will you do to address this issue? I only used approved templates. If they are not considered polite why have them?</td>
<td>0.650, 157</td>
</tr>
<tr>
<td>A0 AM-NEG V A1</td>
<td>Now why would you do that when he specifically stated that he didn't want to discuss the matter for 48 hours and I specifically stated that people should come here? Did you not read his request? Let me ask you this. Assuming we take it that the player might have chosen any door to start with and the host may have opened any unchosen goat-hiding door why do you not insist that the starting sample space must include all the possible door combinations?</td>
<td>0.760, 58</td>
</tr>
</tbody>
</table>

Table 9: Impolite SRL patterns and their sample requests
### Table 10: Polite SRL patterns and their corresponding sample requests

<table>
<thead>
<tr>
<th>SRL Pattern</th>
<th>Sample Requests</th>
<th>Precision, # of Requests containing the Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM-MOD A0 V A1</td>
<td>I was in doubt too but when dealing with such incomprehensible concoctions I prefer to err on the side of BLP caution. Would you like it userfied? I tried to create a similar map for Canada but couldn't get a usable output from the site you used for the Australia map. Could you create a Canadian one?</td>
<td>0.764, 331</td>
</tr>
<tr>
<td>A0 AM-MOD V A1</td>
<td>When I wrote &lt;url&gt; I was not quite sure if it was a synonym of &lt;url&gt; and hence did not perform a merge. Could you clarify if they are indeed the same and if so whether we should merge the articles? Sure thing you've been doing great work so far. Would it help to maybe translate the whole article first as it is and then over the course of time you can play around with the sections and add stuff that you find?</td>
<td>0.571, 319</td>
</tr>
<tr>
<td>AM-ADV A0 AM-MOD V</td>
<td>I hope this makes my concerns clearer. Perhaps we can work together to resolve them? If you're interested I could provide you with Subversion (which is superior to CVS) for VandalProof. Interested?</td>
<td>0.628, 51</td>
</tr>
</tbody>
</table>
Chapter 5: Conclusion and Future work

In this chapter we summarize the main contributions of our work, and discuss its possible extensions and directions for future.

5.1 Contributions

In Chapter 4, we introduced a new set of features to scrutinize politeness from other perspectives besides the unigrams and specific lexical phrases utilized by Danescu-Niculescu-Mizil et al. (2013). First we consider the effect of adding semantic role labels (SRL’s) and/or dependency labels (DL’s) to the unigram model to see if semantic information captured by these labels (SRL’s and DL’s) can help decode politeness in the data. Experimental results (section 4.3) of adding SRL’s or DL’s to baseline model demonstrate improved performance of our politeness classifier in Wikipedia domain, and verify that semantics expressed by these labels not only relate to politeness in data but can also help to decode it. Further, this improvement in the classifier’s performance supports our observation of repeated semantic role label patterns in polite requests and helps discover certain N-gram semantic role label patterns (Figure 5) that act as politeness markers in requests.

In addition, we utilize classification rules to analyze the coupled effects of semantic and lexical information in data to associate with politeness. In the experiments section on
adding rules to our baseline (Table 4.1), we see a rise in the classifier’s performance with SRL’s or DL’s features, although not significantly better than the baseline. However, when we include rules with all other features (SRL’s, DL’s and token count) we get the highest gain for our classifier framework. This result exemplifies when adding rules with other features to our baseline model, each feature has its own contribution to politeness recognition in the data. Also, the results from cross domain analysis (Tables 6) show that these classification rules not only exhibit the highest priority over other feature categories but have stronger associations with the data compared to other attributes.

5.2 Future work

We notice poor performance of our implementation model on Stack Exchange data and in the cross-domain experimental results (Tables 5 and 6). Therefore, it would be important to formulate rules that work for Stack Exchange requests and other domains. In addition, improving the performance of the baseline with SRL’s and/or DL’s features in these domains would be a relevant thing to do.

As discussed previously in the contributions section, classification rules decode politeness more effectively in the data and usually override the effect of all the other feature categories. Thus, there exists a scope of improvement of the classification rules that would identify politeness in data more accurately and boost the classifier’s performance.

Moreover for the future it is reasonable to analyze and compute politeness in other data types such as initiating dialogue, making suggestions or providing responses and not restrict ourselves to requests alone.
Bibliography


Collobert, R. Deep Learning for Efficient Discriminative Parsing, in International Conference on Artificial Intelligence and Statistics (AISTATS), 2011.


Appendix A: Code Implementation of Feature Functions used in Previous Work

In this appendix we provide python code implementation of all the feature functions used in the politeness classifier of the previous work (Danescu-Niculescu-Mizil et al., 2013). In their classifier framework they primary used three categories of feature functions – dependency based, string based and token based feature functions which are related to several politeness strategies of Brown and Levinson, 1987. Python code of these feature functions is as follows –

```python
import os
import re
from itertools import chain
from collections import defaultdict

# Word lists
hedges = [
    "think", "thought", "thinking", "almost",
    "apparent", "apparently", "appear", "appeared", "appears", "approximately",
    "around",
    "assume", "assumed", "certain amount", "certain extent", "certain level", "claim",
    "claimed", "doubt", "doubtful", "essentially", "estimate",
    "estimated", "feel", "felt", "frequently", "from our perspective", "generally", "guess",
    "in general", "in most cases", "in most instances", "in our view", "indicate",
    "indicated",
    "largely", "likely", "mainly", "may", "maybe", "might", "mostly", "often", "on the whole",
    "ought", "perhaps", "plausible", "plausibly", "possible", "possibly", "postulate",
]```

# Positive and negative words from Liu
local_dir = os.path.split(__file__)[0]
pos_filename = os.path.join(local_dir, "liu-positive-words.txt")
neg_filename = os.path.join(local_dir, "liu-negative-words.txt")
positive_words = set(map(lambda x: x.strip(), open(pos_filename).read().splitlines()))
negative_words = set(map(lambda x: x.strip(), open(neg_filename).read().splitlines()))

# Parse element accessors.
# Given parse element string like "nsubj(dont-5, I-4)"
# transform or return specific constituents

parse_element_split_re = re.compile(r"([-\w!?]+)-(\d+)")
getleft = lambda p: parse_element_split_re.findall(p)[0][0].lower()
getleftpos = lambda p: int(parse_element_split_re.findall(p)[0][1])
getright = lambda p: parse_element_split_re.findall(p)[1][0].lower()
getrightpos = lambda p: int(parse_element_split_re.findall(p)[1][1])
remove_numbers = lambda p: re.sub(r"-\d+", "", p)
getdeptag = lambda p: p.split("(")[0]

## Strategy Functions
## Defined as named lambda functions that return booleans
## Each function checks for a single strategy,
## returns True if strategy detected, False otherwise.
## Some functions operate on dependency-parse elements,
## some on string text inputs, some on token lists
####
####
# Dependency-based politeness strategies
please = lambda p: len(set([getleft(p), getright(p)]).intersection(['please'])) > 0 and 1 not in [getleftpos(p), getrightpos(p)]
please.__name__ = "Please"

pleasestart = lambda p: (getleftpos(p) == 1 and getleft(p) == "please") or (getrightpos(p) == 1 and getright(p) == "please")
pleasestart.__name__ = "Please start"

hashedges = lambda p: getdeptag(p) == "nsubj" and getleft(p) in hedges
hashedges.__name__ = "Hedges"

defereence = lambda p: (getleftpos(p) == 1 and getleft(p) in ['great', 'good', 'nice', 'good', 'interesting', 'cool', 'excellent', 'awesome']) or (getrightpos(p) == 1 and getright(p) in ['great', 'good', 'nice', 'good', 'interesting', 'cool', 'excellent', 'awesome'])
defereence.__name__ = "Deference"

gratitude = lambda p: getleft(p).startswith("thank") or getright(p).startswith("thank") or 
"(appreciate, i)" in remove_numbers(p).lower()
gratitude.__name__ = "Gratitude"
apologize = lambda p: getleft(p) in ("sorry", "woops", "oops") or getright(p) in 
("sorry", "woops", "oops") or remove_numbers(p).lower() in ("dobj(excuse, me)", 
"nsubj(apologize, i)", "dobj(forgive, me)")
apologize.__name__ = "Apologizing"

groupidentity = lambda p: len(set([getleft(p), getright(p)]).intersection(['we', 'our', 
"us", "ourselves"])) > 0
groupidentity.__name__ = "1st person pl."

firstperson = lambda p: 1 not in [getleftpos(p), getrightpos(p)] and len(set([getleft(p), 
getright(p)]).intersection(['i', 'my', 'mine', 'myself'])) > 0
firstperson.__name__ = "1st person"

secondperson_start = lambda p: (getleftpos(p) == 1 and getleft(p) in 
("you", "your", "yours", "yourself")) or (getrightpos(p) == 1 and getright(p) in 
("you", "your", "yours", "yourself"))
secondperson_start.__name__ = "2nd person start"

firstperson_start = lambda p: (getleftpos(p) == 1 and getleft(p) in 
("i", "my", "mine", "myself")) or (getrightpos(p) == 1 and getright(p) in 
("i", "my", "mine", "myself"))
firstperson_start.__name__ = "1st person start"
hello = lambda p: (getleftpos(p) == 1 and getleft(p) in ("hi","hello","hey")) or (getrightpos(p) == 1 and getright(p) in ("hi","hello","hey"))
hello.__name__ = "Indirect (greeting)"

really = lambda p: (getright(p) == "fact" and getdeptag(p) == "prep_in") or remove_numbers(p) in ("det(point, the)","det(reality, the)","det(truth, the)") or len(set([getleft(p), getright(p)]).intersection(["really", "actually", "honestly", "surely"])) > 0
really.__name__ = "Factuality"

why = lambda p: (getleftpos(p) in (1,2) and getleft(p) in ("what","why","who","how")) or (getrightpos(p) in (1,2) and getright(p) in ("what","why","who","how"))
why.__name__ = "Direct question"

conj = lambda p: (getleftpos(p) == 1 and getleft(p) in ("so","then","and","but","or")) or (getrightpos(p) == 1 and getright(p) in ("so","then","and","but","or"))
conj.__name__ = "Direct start"

dbtw = lambda p: getdeptag(p) == "prep_by" and getright(p) == "way" and getrightpos(p) == 3
dbtw.__name__ = "Indirect (btw)"

secondperson = lambda p: 1 not in (getleftpos(p), getrightpos(p)) and len(set([getleft(p), getright(p)]).intersection(["you","your","yours","yourself"])) > 0
secondperson.__name__ = "2nd person"

####
# Dependency-based request identification heuristics

polar_set = set(["is", "are", "was", "were", "am", "have", "has", "had", "can", "could", "shall", "should", "will", "would", "may", "might", "must", "do", "does", "did", "ought", "need", "dare", "if", "when", "which", "who", "whom", "how"])
initial_polar = lambda p: (getleftpos(p)==1 and getleft(p) in polar_set) or (getrightpos(p)==1 and getright(p) in polar_set)
initial_polar.__name__ = "Initial Polar"

aux_polar = lambda p: getdeptag(p) == "aux" and getright(p) in polar_set
aux_polar.__name__ = "Aux Polar"

####
# String-based politeness strategies
# (i.e., input is a sentence)

# Verb moods
subjunctive = lambda s: "could you" in s or "would you" in s
subjunctive.__name__ = "SUBJUNCTIVE"

indicative = lambda s: "can you" in s or "will you" in s
indicative.__name__ = "INDICATIVE"

####
# Token list politeness strategies

has_hedge = lambda l: len(set(l).intersection(hedges)) > 0
has_hedge.__name__ = "HASHEDGE"

has_positive = lambda l: len(positive_words.intersection(l)) > 0
has_positive.__name__ = "HASPOSITIVE"

has_negative = lambda l: len(negative_words.intersection(l)) > 0
has_negative.__name__ = "HASNEGATIVE"

####
# strategy_fnc application helper

# For debugging, prints exceptions
VERBOISE_ERRORS = False

def check_elems_for_strategy(elems, strategy_fnc):
    # given a strategy lambda function,
    # see if strategy present in at least one elem
    for elem in elems:
        try:
            testres = strategy_fnc(elem)
            if testres:
                return True
        except Exception, e:
            if VERBOISE_ERRORS:
                print strategy_fnc.__name__
                print e, elem
        return False
## Feature extraction
## Detect politeness strategies in documents
## by applying strategy fncs.
## Return feature dict

# Define the dependency-based strategies to include:
DEPENDENCY_STRATEGIES = [
    please, pleasestart, btw,
    hashedges, really, deference,
    gratitude, apologize, groupidentity,
    firstperson, firstperson_start,
    secondperson, secondperson_start,
    hello, why, conj
]
# And raw text-based strategies:
TEXT_STRATEGIES = [subjunctive, indicative]
# And term list strategies:
TERM_STRATEGIES = [has_hedge, has_positive, has_negative]

# Use strategies to generate list of all feature names
# lambda function turns strategy function names into feature names
fnc2feature_name = lambda f: "feature_politeness_==%s==" % f.__name__.replace(" ", "_")

POLITENESS_FEATURES = map(fnc2feature_name, chain(DEPENDENCY_STRATEGIES, TEXT_STRATEGIES, TERM_STRATEGIES))

#print POLITENESS_FEATURES

def get_politeness_strategy_features(document):
    """
    :param document- pre-processed request document
    :type document- dict with 'sentences', 'parses',
    and 'unigrams' fields
    {
        "sentences": ["sentence 1", "sentence 2"],
        "parses": [
            ["nsubj(dont-5, I-4)"], ...
        ],
        "unigrams": ["a", "b", "c"]
    }
    """
Returns- binary feature dict

{
    feature_name: 1 or 0
}

Currently return binary features- just checking for presence of a strategy. One could alternatively decide to count occurrences of the strategies.

```python
if not document.get('sentences', False) or not document.get('parses', False):
    # Nothing here. Return all 0s
    return {f: 0 for f in POLITENESS_FEATURES}
```

features = {}

# Parse-based features:
parses = document['parses']
for fnc in DEPENDENCY_STRATEGIES:
    f = fnc2feature_name(fnc)
    features[f] = int(check_elems_for_strategy(parses, lambda p:
        check_elems_for_strategy(p, fnc)))

# Text-based
sentences = map(lambda s: s.lower(), document['sentences'])
for fnc in TEXT_STRATEGIES:
    f = fnc2feature_name(fnc)
    features[f] = int(check_elems_for_strategy(sentences, fnc))

# Term-based features:
terms = map(lambda x: x.lower(), document['unigrams'])
for fnc in TERM_STRATEGIES:
    f = fnc2feature_name(fnc)
    features[f] = int(check_elems_for_strategy([terms], fnc))

return features