Using Meaning Specificity to Aid Negation Handling in Sentiment Analysis

Doreen Hii
Computation of Language Laboratory, UCI

Abstract

Sentiment analysis is an automatic way of classifying emotion (positive, negative or neutral) expressed in written texts. This project focuses on one aspect of sentiment analysis – negation handling, which determines the sentiment of an input with negation words such as "no," "not," and "never." I approached this challenge from the stage of negation resolution, assuming the existence of a working negation scope detector. Negation resolution is the stage where an algorithm decides and applies the adjustments needed to account for the effect of negation. A novel strategy, *meaning specificity*, was proposed, where components of meaning are incorporated into negation handling. The value of the meaning specificity approach was first validated via calculations of information gain and then compared with previous methods of negation resolution in a sentiment analysis pipeline. The meaning specificity approach offered higher information gain over non-semantic approaches; at the same time, it achieved the highest performance when tested on a selected hard subset of product reviews.

1 Introduction

Sentiment analysis is a technique to determine the sentiment (sometimes called *valence* or *polarity*) of text (e.g., a phrase, sentence, paragraph, or document), based on linguistic signals of that sentiment. For example, a company or a politician can determine the overall customer satisfaction of a new product or a new policy by classifying reviews into positive, negative, and neutral categories. The benefits of

performing sentiment analysis go beyond monitoring customer satisfaction; it also serves as a foundation for realistic human-machine interaction generally.

One important aspect of sentiment analysis is negation handling, where negations (e.g., in English, words like "not," "no," "never," and "neither") can dramatically alter the *affirmative* expression, i.e., any expression that is not negated. For example, "I am happy" is an affirmative expression with a positive sentiment score while "I'm not happy" is a negated expression with a negative sentiment score. This illustrates one of the many interpretations of negation: negation flips or inverts positive sentiments to negative. Another interpretation of negation is that negation diminishes an affirmative sentiment, as in the case "It's not terrible," where the affirmative context "terrible" is diminished in negativity, though it remains negative after negation is applied. Yet another possible effect of negation is shifting the affirmative sentiment to neutral. For example, "This is bad" carries a negative sentiment while "This is not bad" is somewhat neutral: it is not at negative as before negation is applied but it is not positive enough to be deemed as a positive expression. Notably, humans are extremely sensitive to the nuances of negation and often arrive at the intended polarity. In fact, due to the sophistication of human negation handling, improper machine negation handling stands out to human eyes as quite noticeable.

Importantly, negation often disrupts state-of-the-art sentiment analysis. As an example of this kind of mistake, Google's sentiment analyzer categorized the following statement as a strongly positive message (score=+0.8, where $-1 \leq \text{score} \leq 1$): "I didn't think that the instructions provided were helpful to me." To humans, this sentence clearly expresses a negative sentiment. Given this, I focus here on improving negation handling in sentiment analysis.[1]

Generally, there are two steps to negation handling in sentiment analysis: (1) *negation scope detection* and (2) *negation resolution*. Negation scope detection is first applied to determine the content to be negated, called the scope of negation, and negation resolution adjusts the sentiment in the scope to reflect negation. To date, different approaches to both negation scope detection and negation resolution have been implemented to increase the accuracy of sentiment analysis when negation is involved. My work centers on developing an accurate and lightweight negation resolution algorithm, which would be applied at the second stage of negation handling.

[1]Note that my work focuses on explicit designs of negation handling functions using a lexicon-based approach, which is different than the implicit machine learning approach implemented by Google's sentiment analyzer. However, insights gained from this study can be incorporated into future works in both lexicon-based and machine learning systems.

I propose a novel approach to negation resolution, one that incorporates components of *meaning* in addition to sentiment value. In particular, I focus on one aspect of meaning: *specificity* (the notion that "beautiful" is more specific than "nice"), as one of the many cognitive ingredients utilized by humans during negation handling. I refer to the novel method as the *meaning specificity* approach.

The paper is organized as follows: I first review previous approaches to negation resolution, and then both motivate and distinguish the meaning specificity approach from other approaches. I then discuss measures that may correlate with meaning specificity and how I calculate them automatically for any word or phrase. Following that, I demonstrate the validity of the extracted meaning specificity components by quantifying information gain as a reduction of entropy. I subsequently discuss the incorporation of this approach into a full sentiment analysis pipeline and its evaluation on a development corpus that contains many examples of negated content. I then present the improvement achieved from implementing the meaning specificity approach and highlight the domain in which its application leads to better performance. Lastly, I discuss the limitations and motivate future research in light of the current results.

2 Compilation of dictionaries for lexicon-based sentiment analysis

There are two main approaches to sentiment analysis: lexicon-based and machine-learning-based. A lexicon-based system explicitly specifies the values of sentiment as well as the rules to combine them; a machine learning system implicitly arrives at its predictions after a training phase with lots of examples. In this study, I adopted a lexicon-based approach due to the interpretability inherent to an explicit system and the ability to make direct inferences about the contributions of a specific variable. Using lexicon-based systems, I can isolate the benefits obtained from adding a specific variable, in this case meaning specificity, in improving the performance of a sentiment analyzer. The results of this study can be translated into machine-learning based systems by including meaning specificity as a feature or using word representations that captures the notion of specificity.

The starting point of a lexicon-based sentiment analysis system is their lexicon or dictionary. A sentiment dictionary serves as a lookup table that associates small tokens or building blocks of language (typically a word or a short phrase) with its sentiment value. Sentiment analysis of larger inputs (sentences, paragraphs,

or documents) are realized by aggregating these tokens. Since all components of lexicon-based sentiment analysis hinge on having a reliable and comprehensive dictionary, I discuss in the following subsections strategies that I used to compile my sentiment dictionary.

2.1 A reliable and comprehensive sentiment dictionary for lexicon-based approaches

For a lexicon-based sentiment analysis, it is crucial to have a comprehensive dictionary of sentiment scores for both individual words (e.g., "good") and larger idioimatic phrases (e.g., "good grief"). In an ideal case, a sentiment dictionary would have sentiment values associated with every word that carries sentiment. More importantly, each entry in the sentiment dictionary should be reliable, being a close estimate of reality. In short, there are two crucial properties to a sentiment lexicon: (1) reliability and (2) comprehensiveness.

The most reliable dictionaries publicly available are manually annotated ones (Kiritchenko and Mohammad, 2016a,b). However, there is a trade-off between reliability and comprehensiveness. Due to the cost of manual annotation, these dictionaries are limited in size. In an attempt to create a comprehensive dictionary, Kiritchenko et al. (2014) leveraged text from Twitter and automatically associated the sentiment.

Since manually annotated and automatically generated dictionaries complement each other in comprehensiveness and reliability, I compiled both types of dictionaries into a single dictionary. To limit the compromise on reliability, more reliable (manually annotated) dictionaries were checked for a sentiment score before less reliable (automatically generated) dictionaries, according to the priority order in Table 2.1. Manually annotated dictionaries always ranked higher in priority than automatically generated ones. I further determined the priority ordering of the dictionaries by manually inspecting a random sample of entries and seeing if they accorded with my intuitions about term sentiment scores. This allowed the most reliable entries from the manually annotated lexica to be utilized, but also keep the vast scope of entries available from automatically annotated lexica.

The compiled sentiment dictionary is composed of 248,646 entries of 75,959 unigrams and 172,687 bigrams. Of all entries combined, I compiled a total of 104,034 positive sentiment entries and 144,612 negative sentiment entries.

Table 1: Priority ordering of manually-created and automatically-generated sentiment dictionaries.

	No.	Valence Dictionary	Entries	Source
Manual	1	SemEval2015-English-Twitter-Lexicon	1,500	Kiritchenko et al. (2014)
	2	SCL-NMA	3,200	Kiritchenko and Mohammad (2016a)
	3	SCL-OPP	1,200	Kiritchenko and Mohammad (2016b)
Auto	4	NRC-Hashtag-Sentiment-Lexicon-v1.0	370,660	Kiritchenko et al. (2014)
	5	NRC-Emoticon-Lexicon-v1.0	740,166	Kiritchenko et al. (2014)
	6	NRC-Hashtag-Sentiment-AffLexNegLex-v1.0	227,303	Kiritchenko et al. (2014)
	7	NRC-Emoticon-AffLexNegLex-v1.0	329,315	Kiritchenko et al. (2014)

2.2 Constructing the gold standard with affirmative and negated sentiment scores

I now describe the construction of the gold standard, i.e., a sample lexicon, with examples of affirmative sentiment scores and their negated ones, that would serve as training data to allow tuning of any model parameters. Affirmative sentiment score is the shared starting point for all models, while the negated sentiment scores are the known finishing points. Different models decide the importance of any given feature based on its contribution in moving the model output towards the final target.

I extracted 1631 words (unigrams only) with both affirmative ("happy") and negated ("not happy") sentiment scores from an automatically generated lexicon, the NRC Hashtag Negated Context Sentiment Lexicon **(NRC lexicon)**[2] (Zhu et al., 2014).

Since the lexicon was automatically generated from real-world tweets, it is inherently noisy. To ensure the quality of my gold standard, I applied rigorous

[2]This lexicon was chosen as it is an updated version of Sentiment140 Lexicon (Mohammad et al., 2013), the lexicon which inspired my hypothesis.

filters. As I am interested in words that carry sentiment, I greatly reduced the number of entries for consideration by removing entries whose sentiment value falls in the range -0.01 to +0.1 (close to neutral). I then removed entries where the word is not found in my compiled dictionary from section 2.1. This removed novel words, atypical typos, and entries that contained pure numbers or symbols. Then, I compared the affirmative score from my dictionary with the negated score from the NRC lexicon, and removed instances where the negated sentiment is not in the opposite direction of its affirmative score. This removed random noise from the lexica, e.g., "not great" being more positive than "great", or "not terrible" being more negative than "terrible." To make sure the gold standard represents actual usage of negation, I removed instances of words that are rarely used in negation. This filter removes an entry if the word does not co-occur at least once with any negation words in the 82.8-million Amazon review corpus (He and McAuley, 2016; McAuley et al., 2015). Note that negation is determined by the presence of negation keywords provided in Table 2.

Table 2: Negation words as suggested by Carrillo-de Albornoz and Plaza (2013)

no	not	n't
cannot	never	none
nothing	neither	nowhere
nobody		

The total number of entries after the filtering was 232, with 134 nouns, 44 verbs, 33 adjectives and 9 adverbs. I then only included adjectives and adverbs in my gold standard dictionary. Nouns and verbs were removed because negation of these syntactic categories depends primarily on contextual information, which is out of the scope of the current research. For example, for the noun "egg," "no egg" could carry a negative sentiment when discussing an egg sandwich order but would carry positive sentiment for a vegan restaurant. Similarly, verbs such as "charged" carry positive sentiment to electronic devices while being negative when used in the context of bills.

Among the 42 entries of adjective and adverbs are 34 positive and 8 negative entries. I note that the unbalanced nature of my gold standard is disadvantageous for methods tuned by using the gold standard. One may tweak the filtering criteria but I felt it necessary to apply strict filtering given the noisy data set I am dealing with.

The resulting list of adjectives and adverbs were manually inspected to make sure that they match my intuition of approximate sentiment. A complete list of

words included in the gold standard are in Appendix A. I acknowledge that it is sub-optimal to regard an automatically generated lexicon as the gold standard. However, as a preliminary check for the effect of including meaning specificity in negation handling, I consider the current choice of lexicon reasonable (after applying strict filters and inspecting the entries manually). Future research could consider obtaining human data for a cleaner and more accurate representation of human negation handling, removing pollution of undesirable noise sprinkled across automatically generated datasets.

3 Previous negation resolution approaches and their implementations

3.1 Prior approaches

3.1.1 Inverting

Inverting the sentiment score is the simplest and coarsest way to handle negation. For example, the adjective "good" has a sentiment score of +0.66 and "terrible" has a sentiment score of -0.70. While it seemed reasonable to invert a positive term, i.e., assigning "not good" to have sentiment score of -0.66; inverting a negative term is less intuitive. That is, using inversion to resolve the negated sentiment of "not terrible" would lead to "not terrible" being more positive (0.70) than "good" (0.66), which does not seem to be true.

This highlights one salient problem with using inversion to handle negation resolution. In particular, negating negative valence items like "terrible" seems to require a different process than negating positive valence items like "good". As Reitan et al. (2015) acknowledged, "just inverting the sentiment polarity of a negated term is incorrect." I used inversion as one of my baselines with respect to handling negation resolution.

3.1.2 Asymmetrical shift

One response to the noted asymmetry when negating positive and negative terms is to use an asymmetrical shift. In this approach, there are two sets of rules to resolve negation, one for positive sentiment terms and one for negative sentiment terms. Socher et al. (2013) implicitly implemented asymmetrical shift by learning

separate negation resolution rules for positive sentences and negative sentences, and thereby increased sentiment accuracy.

I used explicit asymmetrical shift as a second baseline with respect to handling negation resolution. To infer the negation handling rules that would apply to positive versus negative sentiment terms, I used the lexicon from Section 2.2 of 42 adjectives and adverbs (8 negative and 34 positive) that was automatically generated from Kiritchenko et al. (2014), which included sentiment scores for both the word (e.g., "happy") and its negated form (e.g., "not happy"). I then performed a multiple regression on this lexicon, yielding the following rule:

$$Negated = -0.06592 - 0.3632 * Sentiment \tag{1}$$

$Sentiment$ is the affirmative sentiment score for the original item (e.g., for "happy" or "terrible") while $Negated$ is the score for the negated item (e.g., for "not happy" or "not terrible"). So, this rule takes the original sentiment score, diminishes it (scaling it by a value of 0.3632), inverts the diminished sentiment score, and then shifts the final negated sentiment by a constant (-0.06592). Notably, the constant (-0.06592) causes an asymmetry in positive vs. negative sentiment terms. For example, an original score of 0.6 leads to a negated score of -0.28 (shift amount: 0.88), while an original score of -0.6 leads to a negated score of 0.152 (shift amount: 0.75). So, positive valence terms have a larger shift compared to negative valence terms. This means negative terms are more likely to be less positive (and possibly even remain negative) when negated, in line with Kiritchenko et al. (2014). Using the equation, "not good" is assigned a negated sentiment score of -0.30 and "not terrible" a negated sentiment of 0.19; so, this approach yields more intuitive negated sentiment scores compared to inverting. However, the effectiveness of applying the same equation to all negation instances is yet to be determined. With an appropriate shifting equation, asymmetrical shift is a straightforward and simple approach. Nevertheless, accuracy when applying a fixed rule to every negated term is yet to be determined.

3.1.3 Antonym dictionary

Another approach is to use an antonym dictionary (Carrillo-de Albornoz and Plaza, 2013), which recognizes that a term's sentiment score may not capture all the components necessary to compute its negated sentiment score. An antonym dictionary works by first replacing the to-be-negated word with its antonym (e.g., "not good" replaced with "bad"). Then, the sentiment value of the antonym (e.g.,"bad")

will be used as the negated sentiment for the original term (e.g., "good"). With an antonym dictionary negation approach, "not good" would have a sentiment of -0.5 (because "bad" has a sentiment score of -0.5) and "not terrible" would have a sentiment of 0.66 (because "good" has a sentiment score of 0.66[3]). This approach therefore requires an antonym dictionary in addition to a sentiment score dictionary. However, to my knowledge, a robust antonym dictionary doesn't yet exist and the construction of one requires substantial work.

I used an antonym dictionary approach as a third baseline with respect to negation resolution. As mentioned, one significant issue is reliably identifying a word's antonym. To identify antonyms for sentiment-bearing words, my implementation attempts to extract antonym relationships from WordNet (Miller, 1995). If a target term's antonym is not directly defined by WordNet, a sequence of searches is performed before I conclude a word has no antonym. Take for example the verb "recommend." At first pass, there is no antonym directly defined for "recommend" at WordNet. Therefore, I iterate through the target's synonyms to identify the first defined antonyms. Synonyms of "recommend" are "urge" and "advocate." However, those two words do not have direct antonyms either. I proceed to check if any of the target term's attributes have antonyms. In the case of "recommend," there is no attribute attached to it. Since that failed, I then try the term's derivationally-related forms (words that share the same root and are semantically related (Miller, 1995)), which gives us "recommendation" and "urgency," both of which did not have antonyms associated. Lastly, I move to the listed similar words, which unfortunately for "recommend," is an empty list. Since all searches failed to identify an antonym, I conclude that the word "recommend" does not have an antonym. In the current implementation, I returned the original sentiment score "recommend" if it exists in my sentiment dictionary, 0 otherwise. Future research could explore diminishing the sentiment score of an antonym by a constant before outputting the prediction (Carrillo-de Albornoz and Plaza, 2013).

[3]Note that the direct antonym of "terrible" is "unalarming." However, because the term "unalarming" does not exist in my sentiment dictionary, the indirect antonym of "terrible" became "good" via the closest related word "bad".

4 Motivation: Consideration of content and context

4.1 Base sentiment score isn't enough

With the exception of the antonym dictionary approach, lexicon-based strategies often handle negation without considering the content being negated – instead, only the original affirmative sentiment score is used to determine the negated sentiment score. Yet, words within the same polarity (e.g., positive, like "nice" and "beautiful") may shift towards the opposite polarity with different intensities (e.g., "not nice" seems much worse than "not beautiful").

Kiritchenko et al. (2014) pointed out some intuitive examples of this kind, shown in Table 3. In the extreme cases, words with similar sentiment scores (e.g., "beautiful" and "nice" have similar positive affirmative polarity around 1; "bad" and "shame" have similar negative affirmative polarity around -1.3) had very different negated sentiment scores. In this case, the negated sentiment score for "beautiful" was 0.217 while the negated sentiment score for "nice" was -0.912; the negated sentiment score for "shame" was -0.722 while the sentiment score for "bad" was 0.021.

Term	**Sentiment140 Lexicons**	
	Affirmative	**Negated**
Positive terms		
great	1.273	-0.367
beautiful	1.112	0.217
nice	1.149	-0.912
good	1.167	-1.414
Negative terms		
terrible	-1.850	-0.890
shame	-1.548	-0.722
bad	-1.674	0.021
ugly	-0.964	-0.772

Table 3: Cases where words with similar original sentiment scores received dissimilar negated scores. Data from *Table 3: Example sentiment scores from the Sentiment140 Base, Affirmative Context (Affirmative) and Negated Context (Negated) Lexicons* (Kiritchenko et al., 2014).

Neither inverting nor asymmetrical shift can handle this negation behavior, as both would cause "beautiful" and "nice" to have similar negated scores, and "shame"

and "bad" to have similar negated scores. Antonym dictionaries are a step towards including more than original sentiment score information, but are still imperfect. In particular, one limitation of antonym dictionaries is how they deal with negative sentiment words; for example, the phrase "not awful" is not equivalent to its antonym "wonderful" (Ruytenbeek et al., 2017), though an antonym dictionary would yield scores that assume this to be the case.

This motivates us to investigate a new method of negation handling, taking inspiration from how humans seem to process negated sentiment. On the basis of examples like those in Table 3, I considered what might distinguish "beautiful" from "nice," and "shame" from "bad." It seemed to us that "beautiful" is more specific than "nice," and "shame" is more specific than "bad." So, I hypothesized that *meaning specificity* could impact negated sentiment. One way to concretely think about meaning specificity is the range of contexts that a word may be used in. More specific words, such as "beautiful" and "shame," are appropriate in fewer contexts while less specific words, such as "nice" and "bad," can be used in many more contexts. Based on the concrete examples I had available, it seemed that negating words with a more specific meaning involves less shifting towards the opposite sentiment direction (i.e., "not beautiful" is less negative than "not nice;" "not a shame" is less positive than "not bad"). Therefore, taking into consideration meaning specificity would allow the subtle differences when negating words with similar affirmative sentiment to be captured.

4.2 How could we tell if meaning specificity was useful?

To assess whether a meaning specificity approach could be useful, it's helpful to consider negation resolution in information-theoretic terms, specifically *information gain*, which can be interpreted as the increase in predictability of one random variable by additionally observing another random variable. In this context, the random variable of interest is the negated sentiment scores given the affirmative scores $N|A$, and the observed random variable for potential information gain is the meaning specificity values M. Intuitively, I expect to be more certain about the negated sentiment of a word if both its affirmative and meaning specificity value are known (i.e. $N|A, M$) than just knowing its affirmative value (i.e. $N|A$). If there is information gain, then there is the potential for a negation resolution algorithm to utilize meaning specificity in its prediction. If there is no information gain, then any algorithm taking meaning specificity into account should not perform any better than one that ignores it.

Information gain can be quantified precisely as follows. Consider the affirma-

tive sentiment score of a word $a_{word} \in A$, the negated sentiment score $n_{word} \in N$, and the word's meaning specificity $m_{word} \in M$. A naive negation resolution attempts to predict the negated sentiment n_{word} by computing its sentiment distribution, given the affirmative sentiment score, $P(N|a_{word})$. If the word's meaning specificity m_{word} provides relevant information for predicting the word's negated score n_{word}, a non-zero mutual information I is expected between the two distributions. More generally, if the distribution over negated scores N given the affirmative scores A as well as meaning specificity M provides relevant information for predicting the distribution over negated scores N given just the affirmative scores A, then there should be non-zero mutual information between the distribution over meaning specificity M, and the distribution over negated scores given the distribution over affirmative scores $N|A$. This mutual information is shown below as $I[M : N|A]$.

$$I[M : N|A] = H[N|A] - H[N|A, M]$$

More specifically, $H[N|A]$ is the entropy of N while observing A, and $H[N|A, M]$ is the entropy of N while observing A and M together. Entropy – defined as the average amount of information required to encode the values of a random variable – can be thought of as information content. Low entropy is desirable because it means that a random variable is more predictable, thus requiring less information to encode. Information gain (or mutual information) $I[M : N|A]$, defined as the difference between $H[N|A]$ and $H[N|A, M]$, which can be understood as the amount of relevant information provided in predicting N given A when M is also observed. Because any real signal can only increase certainty (i.e. decrease entropy), it follows that $H[N|A, M] \leq H[N|A]$. If $H[N|A, M] = H[N|A]$, then mutual information between $N|A$ and M is zero, and there is no relevant information to be gained from M. We are interested in $H[N|A, M] < H[N|A]$, meaning that mutual information is non-zero and M is a real signal that provides relevant information.

As an initial demonstration of concept, a toy lexicon of 10 words was constructed. The naïve baseline approach takes the affirmative sentiment of a word and tries to predict its negated sentiment. Affirmative sentiment scores are extracted from the compiled sentiment dictionary while the negated sentiment is determined by the NRC lexicon with scores normalized (Zhu et al., 2014). For illustrative purposes, both affirmative and negated sentiments are categorized into three sentiment classes (positive, neutral, and negative). To evaluate the benefits of including meaning specificity, each entry in the toy lexicon is also

assigned a specificity level (labeled as not specific, somewhat specific and very specific) subjectively determined by the author. Calculation for information gain I is performed, looking specifically for non-zero values $H[N|A, M] < H[N|A]$. $I[M : N|A] = H[N|A] - H[N|A, M]$ is calculated as in 4 below.

$$
\begin{aligned}
H[N|A] &= \mathbb{E}[\log \frac{1}{p(n|a)}] \\
&= \sum_{a \in A, n \in N} p(n, a) \cdot log \frac{1}{p(n|a)} \\
H[N|A, M] &= \mathbb{E}[\log \frac{1}{p(n|a, m)}] \\
&= \sum_{n \in N, m \in M, a \in A} p(n, a, m) \cdot log \frac{1}{p(n|a, m)}
\end{aligned} \tag{2}
$$

Each entropy term is the expectation of information content over its distribution ($log \frac{1}{p(n|a)}$ and $log \frac{1}{p(n|a,m)}$ over $N|A$ and $N|A, M$ respectively). The probability terms are calculated by counting the relative frequency of words with the particular combination of n, a, and/or m categories.

As a sanity check, information gain values were calculated for a model observing meaning specificity values as random noise sampled from a uniform distribution (not specific, somewhat specific, and very specific) on top of observing affirmative values. This is asserting that the model of interest, i.e. model that takes in affirmative values and real signal of meaning specificity, would still have higher information gain compared to the sanity check model.

This seems to indeed be true: averaging over 1000 trials, the information gain for this sanity check model ($I[M' : N|A]$ using random meaning specificity samples) is 0.36 bits. Information gain ($I[M : N|A]$ using the subjectively determined meaning specificity values is 0.50 bits. There is a $1.4\times$ increase in information gain from using values of M over uniformly random values. So, meaning specificity adds value by encoding more relevant information to aid prediction of negated sentiment scores. This motivates the subsequent exploration of concretely quantifying meaning specificity to be utilized as a feature in negation resolution.

4.3 A first attempt to incorporate meaning specificity

As a first attempt to assess meaning specificity in an automatic and easy-to-implement way, I considered aspects of a term's usage that might be correlated

Table 4: A manually created toy lexicon of 10 words. Affirmative and negated sentiment scores extracted from compiled sentiment dictionary 2.1 and NRC lexicon (Zhu et al., 2014) respectively. Meaning specificity values were subjectively determined by the author.

Words	**Affirmative**	**Negated**	**Specificity**
happy	Positive	Negative	Not
prepared	Neutral	Neutral	Very
good	Positive	Negative	Not
equal	Neutral	Neutral	Very
disappointed	Negative	Positive	Somewhat
easy	Positive	Neutral	Not
angry	Negative	Positive	Not
positive	Positive	Neutral	Not
best	Positive	Neutral	Somewhat
truly	Positive	Neutral	Very

with its specificity. The first aspect being considered is frequency, with the idea that more-specific terms might be used less frequently than less-specific terms – this is because the more-specific terms are appropriate in fewer contexts. As a second related aspect, I considered inverse dispersion (Gries, 2008), which is a measure of the diversity of contexts a term appears in. Inverse dispersion captures the different contexts a word may be used, which ties more directly to my intuitive sense of meaning specificity. The idea would then be that more-specific terms appear in less diverse contexts than less-specific terms.

We can interpret the interaction between frequency and inverse dispersion as capturing a similar concept to term frequency-inverse document frequency (tf-idf) (Salton and McGill, 1986). Tf-idf is commonly used in search engines to match results with searches, and reflects the importance of a term *t* to a collection of documents. The tf-idf of a term *t* is calculated by multiplying the term's frequency (**tf**) with the term's inverse document frequency (**idf**). Term frequency is calculated as the number of occurrences in a document while inverse document frequency is calculated by the formula below:

$$idf(t) = log_e(\textit{Total number of documents}/\textit{Number of documents with term } t)$$

Generally, high tf coupled with high idf means that term *t* is specific to the topic under discussion, which is useful for identification of relevant search results. An

interaction between frequency and inverse dispersion could be viewed as switching idf into inverse dispersion, yielding another variant of a term specificity metric. While I do not investigate this metric here, it is an interesting avenue for future work.

5 Implementation of meaning specificity

I now discuss the estimation of each component I use to approximate meaning specificity: frequency and inverse dispersion. I also highlight the use of an accessible corpus to quickly approximate meaning specificity. Information about meaning specificity is stored in a lookup table where the keys correspond to a language token. For example, to retrieve the frequency count of the token "good," one would refer to the frequency lookup table and extract the value associated with the key "good." Negation for meaning specificity is handled via an equation with four variables: sentiment score, frequency, inverse dispersion, and the interaction between frequency and inverse dispersion. The weight of each variable is tuned using my gold standard. I demonstrate how to integrate this version of meaning specificity into a sentiment analysis pipeline, including strategies to handle novel tokens that aren't in the lookup table. I subsequently introduce a lightweight version of meaning specificity negation handling strategy which is a variation of my original proposal.

5.1 Estimating term frequency and inverse dispersion

One way to estimate frequency and inverse dispersion for a given term is to approximate components of meaning specificity using any corpus, assuming that the corpus is a reasonable representation of everyday language use. Here, I leveraged 82.8 million reviews retrieved from an Amazon product review corpus (He and McAuley, 2016; McAuley et al., 2015). Each element used for approximating meaning specificity (frequency and inverse dispersion) was calculated from the corpus. I extracted the information for every entry that exists in the sentiment dictionary.

Frequency was calculated by counting the occurrences of the sentiment term in the Amazon product review corpus. Inverse dispersion (**ID**) was calculated as in Gries (2008) (see (3)), and can be viewed as how widespread the word's use is.[4] A

[4]Note that Gries (2008) refers to ID as dispersion. I decided to adopt a different terminology for

word with low inverse dispersion is used equally often in all contexts (i.e., its use isn't very specific).

$$(3) \quad ID_{word} = \frac{\Sigma_{i=1}^{n} |observed_{word_i} - expected_{word_i}|}{2} \quad , i = \#\ corpus\ parts$$

$$observed_{word_i} = \frac{freq_{word_i}}{total\ freq_{word}} \quad , \quad expected_{word_i} = \frac{size\ of\ corpus\ part_i}{size\ of\ entire\ corpus}$$

To calculate inverse dispersion, we need to have different samples that represent different contexts a word could be used in. So, I first divided the Amazon product reviews corpus into 10 equal parts of approximately 8.28 million words each (i.e., i=10). The intuition is that the 10 different corpus sections represent different contexts (e.g., reviews about clothing or reviews about electronics). Then, the observed usage in that corpus part is compared against the expected usage in that corpus part. For the observed usage $observed_{word_i}$, I calculate the frequency of each sentiment word ($freq_{word_i}$). Then, I normalize this frequency by dividing by the word's overall frequency ($total\ freq_{word}$). For the expected usage $expected_{word_i}$, I set this to 0.1. That is, if a word had completely widespread use irrespective of context, it would appear in each of the equally-sized 10 parts at the same rate. So, its expected appearance is $\frac{1}{10}$=$\frac{size\ of\ corpus\ part}{size\ of\ entire\ corpus} \approx 0.1$.)

The difference between the observed and expected usage is then summed in each of the 10 parts, $\sum_{i=1}^{10} |observed_{word_i} - expected_{word)i}|$ and divided by two to change the range of inverse dispersion from [0,2] to [0,1]. So, this calculation, in effect, yields how much the word's observed usage differs from a maximally-dispersed word's usage. The resulting inverse dispersion value ranged from 0 to 1, with 0 indicating that the sentiment term's use wasn't different from a maximally-dispersed word's (i.e., it was distributed evenly throughout the corpus) and 1 indicating the term's usage was very different from a maximally-dispersed word's (i.e., the sentiment term clustered in certain corpus sections and therefore is used in more restricted contexts).

5.2 Constructing a lookup table for meaning specificity

A lookup table was built to allow rapid access of frequency and inverse dispersion for words that exist in my sentiment dictionary. During negation resolution, the

a more intuitive interpretation: an ID of 0 means that the word is distributed across all contexts; an ID of 1 indicates that the word clusters in a specific context.

to-be-negated token is the key to search for its matching frequency and inverse dispersion values. If the key exists in the dictionary, the corresponding values for meaning specificity were used in subsequent calculations.

5.3 A negation transformation incorporating meaning specificity

The frequency and inverse dispersion values were incorporated into a function that would take a term's original sentiment score and generate a negated sentiment score. I used a linear equation relating the original score, the term's frequency, the term's inverse dispersion, and the interaction between frequency and inverse dispersion to the term's negated score. To determine appropriate weights for this linear equation, I used the same 42-word list that was used to implement my version of asymmetrical shift (see Section 2.2 and Table 10). More specifically, I performed a multiple regression with these 42 original and negated scores, along with the accompanying frequency and inverse dispersion values. This yielded the equation in (4), where $Frequency$ is the raw term frequency and $InverseDispersion$ is calculated via the metric outlined in Section 5.1.

(4)
$$\begin{aligned} Negated = {} & (-6.112665 \times 10^{-2}) - 0.3851552 * Original \\ & + (7.751644 \times 10^{-9}) * Frequency - 2.26025 * InverseDispersion \\ & - (1.976151 \times 10^{-6}) * Frequency * InverseDispersion \end{aligned}$$

To allow easy interpretation on the weights, I retrained the multiple regression model with frequency normalized. Note that this version of the equation is only used to interpret the weights by restricting both frequency and inverse dispersion in the same range of [0,1]. The equation taking raw frequency as input was used for actual sentiment calculation.

(5)
$$\begin{aligned} Negated = {} & -0.061 - 0.39 * Original \\ & + 2.77 * Frequency - 2.26 * Dispersion \\ & - 705.61 * Frequency * Dispersion \end{aligned}$$

Interestingly, based on these inferred weights, the interaction of frequency and inverse dispersion matters far more (by two orders of magnitude) than either component individually. Future work can compare the effects between applying frequency and inverse dispersion combined with equations applying td-idf.

5.4 How to handle novel tokens: Inferring frequency and inverse dispersion from sentiment scores

While unigrams and bigrams appearing in my constructed dictionary will have their meaning specificity components pre-calculated from the Amazon product review corpus, I will also encounter terms or larger phrases that I need meaning specificity estimates for. While these terms or larger phrases will have sentiment scores associated with them, they won't have frequency or inverse dispersion values readily available. Having a way to infer those meaning specificity components on the basis of a term's original sentiment score will allow for a more robust negation resolution algorithm, as sentiment scores are easily accessible at any stage in a sentiment analysis pipeline.

As an illustration, consider the sentence "I am not happy about it" with the negation scope "happy about it." In this case, there is only 1 sentiment-carrying term "happy" in the scope. Therefore, the meaning specificity component for the to-be-negated phrase "happy about it" boils down to the negation of "happy." Lookup tables for meaning specificity components would work in this case. However, consider another review sentence "I will not recommend anyone to rely on it," where the negation scope ("recommend anyone to rely on it") encompasses multiple sentiment-carrying terms ("recommend" and "rely"). In this example, the meaning specificity component of the to-be-negated phrase cannot be determined easily. In this case, the sentiment of the to-be-negated phrase would first be aggregated. Then, the aggregated sentiment would be used to infer its meaning specificity components.

To infer a term's frequency t_f and inverse dispersion t_{id} on the basis of its original sentiment score $t_{s_{orig}}$, I divided all existing original sentiment scores t_s into discrete bins of size 0.04. For example, a bin included terms with -1.0$< t_s <$-0.96, and the following bin included terms with -0.96$< t_s <$-0.92. There were a total of 52 bins, with the 1st and 52nd bins containing only those terms with t_s=-1.0 and t_s=-1.0 respectively, the extreme sentiment values.

For each sentiment bin, mean frequency value and mean inverse dispersion value were calculated, along with their standard deviations. The resulting mean and standard deviation parameters were used to reconstruct a Gaussian distribution of meaning specificity components for any term with a sentiment score falling within that bin. So, the general procedure when encountering a novel term or larger phrase is the following:

- Access its sentiment score $t_{s_{orig}}$

- Determine the bin of that sentiment score
- Access the Gaussian distribution of frequency and inverse dispersion values that was pre-computed on the basis of the mean and standard deviation of those values for that bin
- Draw a random sample from the distribution for each component (i.e., one draw for the frequency value t_f and one draw for the inverse dispersion value t_{id})
- Use the sampled frequency and inverse dispersion values in the equation (4)

I note that this method can't handle the cases where words of similar sentiment values have different negated scores. Instead, it's a back-off alternative for terms and phrases that aren't in my sentiment dictionary, and so don't have pre-computed meaning specificity components.

5.5 Further exploration: Lightweight variation of meaning specificity negation handling approaches

My proposed meaning specificity negation handling approach utilizes two components: a meaning specificity lookup table, and a back-off method of inferring meaning specificity components, with priority given to the lookup table. Therefore, I refer to this approach as *meaning specificity lookup approach*.

In this section, I discuss an alternative to the meaning specificity lookup approach. The variation, which I call *meaning specificity lite approach*, drops the component of the meaning specificity lookup table and always uses the back-off method. Recall that the back-off method infers meaning specificity components from their corresponding distributions (details in Section 5.4). This is a "lite" version of the meaning specificity lookup approach for two reasons: (1) it removes the need for a comprehensive meaning specificity lookup dictionary, and (2) it is easier to construct. By removing dependency on a lookup dictionary, it eliminates the need to compile a dictionary in the first place and therefore eliminates the need to obtain representative frequency and dispersion values for each entry in the dictionary. Second, it is easier to construct distributions of meaning specificity within each sentiment bin than to construct a comprehensive lookup table, because each distribution is a class where entries in a lookup table belongs to. That is, only two distributions are needed to recreate the meaning specificity components, namely frequency and inverse dispersion, for each of the 52 sentiment bins. The

constructed distribution can thereby be applied to all other instances in the sentiment bin, regardless of whether the item has been seen before. Furthermore, construction of those distributions uses bounded and controllable memory: one can choose to stop sampling for a sentiment bin once there are enough entries m to construct the distribution, with the magnitude of m being a free parameter chosen by the user. While greater m approximates the true distribution better, one can afford to set a lower value for m in cases where resources (time or memory) are limited.

For the current study, I simply constructed the underlying distributions using all the entries in my sentiment dictionary. Future research could investigate constructing the distributions by sampling an equal number of entries in each sentiment bin.

6 Validation of Frequency and inverse dispersion: Values added to improve data separability

Section 4 illustrated a proof of concept that there is value in considering the meaning specificity of a token when handling negation. As expected, the original affirmative score is strongly but not perfectly indicative of the negated sentiment. That is, although affirmative sentiment roughly dictates the resulting negated sentiment, it fails to disambiguate cases where words have similar affirmative sentiment but wildly different negated sentiment. By including meaning specificity as an additional feature that provides information gain, (1) I obtained a non-zero information gain when given the additional feature of meaning specificity, $I[M : N|A]$, and (2) it passed the sanity check of $I[M : N|A] > I[M' : N|A]$, where M' are random meaning specificity values sampled from a uniformly random distribution. However, are frequency and inverse dispersion sensible estimates of meaning specificity? Recall that in section 4, utility when considering meaning specificity was demonstrated – but using toy specificity categories (not specific, somewhat specific, and very specific). Here I demonstrate that my implementation of meaning specificity with frequency and inverse dispersion is useful, as indicated by information gain above a baseline.

I performed the same calculations with my compiled sentiment dictionary and implementation of meaning specificity. Consistent with my hypothesis, I find that (1) the information gain when given meaning specificity $I[M : N|A]$ is again non-zero (0.018 bits) and (2) is greater than the random baseline information

gain $I[M' : N|A]$ (0.0043 bits). In other words, my implementation of meaning specificity provides 4.2 times more information than random values.

To better understand the contributions of each feature (affirmative scores, frequency, inverse dispersion, and the interaction between frequency and inverse dispersion) to the final classification of negated sentiment, I implemented a decision tree as a data exploration technique. Based on the features provided, a decision tree ranks the features based on their importance to maximally reduce entropy at each node (Loh, 2014). Features are used in the decision process in order of their precedence: the feature at the root is the most distinguishing feature, and that which offered the most information gain.

Two decision tree classifiers of maximum depth of three were built to compare the information gain (1) when including meaning specificity features, DT_{A+}, and (2) with the model considering affirmative scores alone, DT_A. A limit was placed on the maximum depth for both decision trees because without the constraint, the optimal choice for any decision tree would be to partition each point of classification into its own node, which essentially reduces the tree to a look-up table. The choice of maximum depth 3 is arbitrary, as it was the suggested depth of the module[5] and it seems reasonable for an exploratory analysis of feature importance. The features of the decision tree were the actual values of affirmative sentiment (real numbers in [-1,1]) extracted from my sentiment dictionary and/or meaning specificity components obtained from the lookup table. The gold standard dataset was chosen in this exploration for two reasons: (1) it is a manually verified dataset with a known target negated sentiment, and (2) it contains entries that have similar affirmative sentiment but vastly different target negated sentiment, such as the three-way comparison between negated phrases "not best," "not great," and "not nice." The affirmative sentiment scores for "best," "great," and "nice" are fairly similar (.81, .73, .72). However, the negated sentiment scores are different in an intuitive way ("not best" being -.02, "not great" being -.34, "not nice" being -1.9). Target values were the negated sentiment scores of the gold standard (see Appendix A). The problem is further simplified into a classification problem by binning the real value negated sentiment scores into three classes: positive, neutral, or negative, using the scale described in Table 5. For example, "not best" reflects a neutral sentiment, "not great" and "not nice" both reflect negative sentiment.

Figures 1 and 2 show visualizations of the decision trees. Each split of a decision tree informs us of the intermediate steps taken which lead to the final infor-

[5]Module from sklearn.DecisionTreeClassifier (https://scikit-learn.org/stable/modules/tree.html#tree).

Sentiment scores (range)	Sentiment bins
$[-1, -.2)$	Negative
$[-.2, +.2)$	Neutral
$[+.2, +1]$	Positive

Table 5: Mapping from sentiment scores to sentiment bins

mation gain. It also transparently depicts the essential components for calculation of information gain at each node: samples, value, and entropy. Samples refers to the number of samples for consideration under the node, starting with all 42 terms grouped under the root node. Value represents the value bins or classifications, which tells us the correct distribution of the samples in the format [number of positive samples, number of neutral samples, number of negative samples]. Entropy is calculated based on the distributions of the samples in its value bins. For example, referring to Figure 1, the root node has entropy of 1.125, which is obtained through calculation $-\frac{30}{42} * log_2(\frac{30}{42}) - \frac{8}{42} * log_2(\frac{8}{42}) - \frac{4}{42} * log_2(\frac{4}{42})$.

With the essential components, information gain as quantified by reduction in entropy, can be calculated as we progress deeper into the tree by comparing the resulting entropy values before and after a split. For example, in Figure 1, the information gain obtained after splitting at the root node (on the feature "Affirmative score" at threshold 0.017), entropy reduced from 1.125 to $\frac{9}{42} * 0.991 + \frac{33}{42} * 0.439$ = .5591 (weighted entropy from the resulting children nodes after the split); this results in an information gain of $1.125 - .5591 = .5659$ (entropy before split - entropy after split).

Not surprisingly, both decision tree models failed to classify the data perfectly after 3 levels (the final entropy scores are non-zero), with DT_{A+} performing better than DT_A. The sum of the resulting entropy for DT_A is higher ($2 * (.918) + .222 = 2.058$) compared to that of DT_{A+} ($.544$). Looking at the structure of DT_{A+},we can see that meaning specificity components played consequential roles. Features such as inverse dispersion and the interaction between frequency and inverse dispersion were used early in the decision process, supporting the hypothesis that frequency and inverse dispersion are sensible factors for negation handling.

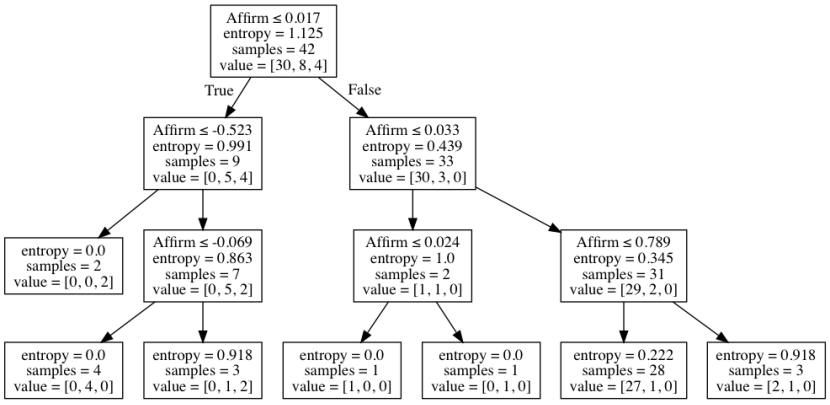

Figure 1: Visualization of DT_A, the decision tree with the affirmative sentiment scores alone as decision features. I indicate affirmative sentiment as "Affirm." Each node specifies the decision criteria, followed by its entropy state before splitting, the number of samples before splitting, and the target classes of those samples [positive, neutral, negative].

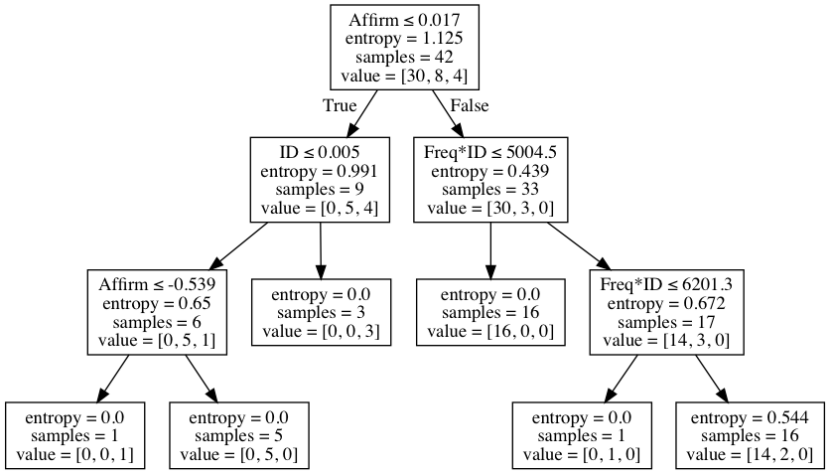

Figure 2: Visualization of DT_{A+}, the decision tree with both the affirmative sentiment scores and meaning specificity components (frequency, inverse dispersion, and the interaction between frequency and inverse dispersion) as features. I indicate affirmative sentiment as "Affirm," frequency as "Freq," and inverse dispersion as "ID." Each node specifies the decision criteria followed by its entropy state before splitting, the number of samples before splitting, and the target classes of those samples [positive, neutral, negative].

7 Incorporating negation resolution approaches into a sentiment analysis pipeline

We have seen now that there is theoretical support for including meaning specificity as a plausible component in negation handling. The following sections present the tests performed on my proposed method in a practical setting of sentiment analysis. In this section, I focus on the supporting parts of a sentiment analysis pipeline.

I first discuss the implementation of an important stage that precedes negation resolution: negation scope detection. Recall from section 1 that negation scope detection identifies the content that needs to be negated. I consider three approaches to negation scope detection, as there's no current consensus on the best negation scope detection approach. I then discuss the next stage after negation resolution, which is the aggregation of negated sentiment. This step defines how multiple sentiment-bearing terms are combined into a single sentiment score. I investigated two sentiment aggregation methods that take in the negated sentiment of tokens and output an aggregated numerical representation.

7.1 Negation scope detection

To arrive at the stage of negation resolution, we first need to detect the content to be negated, i.e., the scope of negation. Importantly, the accuracy of negation scope detection affects the performance of negation handling – in particular, the effectiveness of any negation resolution algorithm is only as good as that of negation scope detection. That is, no matter how good an approach is at resolving negation, it won't work well if the wrong content has been identified as the scope of negation. Therefore, I considered three current approaches to negation scope detection: n-grams (Blair-Goldensohn et al., 2008; Taboada et al., 2011; Thelwall et al., 2012), parse trees (Klein and Manning, 2003; Carrillo-de Albornoz and Plaza, 2013; Socher et al., 2013), and a machine learning based classifier (Enger et al., 2017). I give a brief summary for each approach in the following subsections.

7.1.1 N-grams

One simple and surprisingly effective approach to negation scope detection is to define the negation scope as the n subsequent words following a negation (Blair-Goldensohn et al., 2008; Taboada et al., 2011; Thelwall et al., 2012). For example, in the sentence "I like how it looks but I will not recommend anyone to rely on

it," a 4-gram negation scope detector marks the subsequent four words after "not" ("recommend anyone to rely") as the negation scope. I used the list of negation words in Table 2 to detect negation scopes. In cases where punctuation was reached before the n count was complete, I terminate the negation scope at the punctuation. Because common values for n range between one and four, I combined 1-, 2-, 3- and 4-gram negation scope detectors in pilot analyses with the inversion negation resolution approach and evaluated them on the test dataset described in section 8.2. The best performing n value was 4. I subsequently used 4-grams in combination with other negation resolution methods for all other analyses.

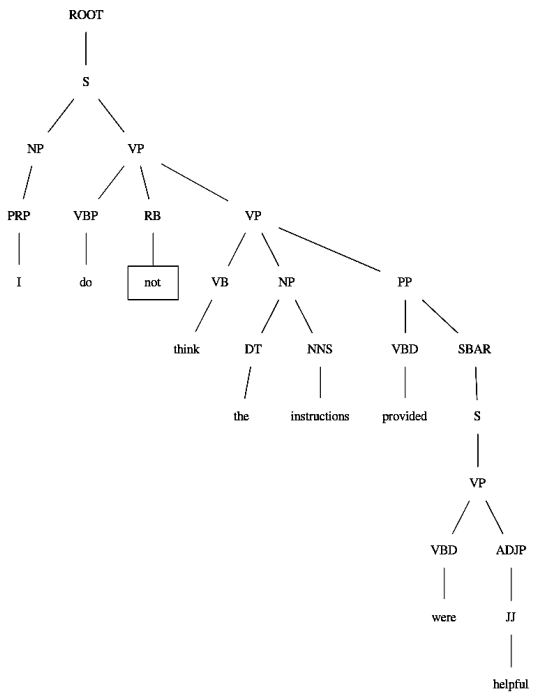

Figure 3: Negation scope as identified by the Stanford-parser-generated parse tree. After the negation word "not," all subsequent siblings and children of those siblings are labeled as in the negation scope.

7.1.2 Parse tree

A linguistically-motivated approach to negation scope detection utilizes the syntactic structure of a sentence, as implemented in its parse tree (Carrillo-de Albornoz and Plaza, 2013; Socher et al., 2013). Once a parse tree is available, the negation scope can be defined as all subsequent siblings and children of the negative node. For the example illustrated in Figure 3, the subsequent sibling of the negation word "not" is the verb phrase (VP). Thus, that VP and all its children are labeled as in the negation scope of "not". Negation was identified with the same list of negation cue words as in Table 2. Prior studies reported promising results using the Stanford Parser to identify the scope of negation. Therefore, I adopted the Stanford Parser in my implementation as well.[6]

[6]I do note, however, that it's imperfect and can generate incorrect parses sometimes (as with Figure 3). However, for the purposes of negation scope detection, it may often be good enough; for example, in Figure 3, the correct scope is all the material after "not" ("think...helpful"), and this is

7.1.3 Machine learning based negation scope classifier

Machine learning has also been used to create more sophisticated tools for negation scope detection. I utilized a pre-trained supervised machine learning strategy called Negtool, which relied on a support vector machine (SVM) (Enger et al., 2017). Specifically, negation cue words were first identified by a SVM. Then, another SVM analyzing for negation scope operates at a sentence level, taking in features such as the word lemma, part of speech, and the relationship between each token extracted from a dependency parser as input. The negation scope detector SVM was trained on a negation annotation corpus, the ConanDoyle-neg corpus (Morante and Daelemans, 2012). The corpus was generated by human annotation on two texts, The Hound of the Baskervilles and The Adventure of Wisteria Lodge. It contains approximately 4000 sentences with around 1000 sentences that contained negation.

7.2 Combining negation scope detection with negation resolution

Since it is currently unknown how negation scope detection interacts with different negation scope resolution approaches, I conducted a systematic comparison of negation scope detection methods in combination with different negation resolution methods. This resulted in 3 negation scope detection approaches (4-gram, parse tree, negtool) by 4 negation scope resolution methods (inverting, asymmetrical shift, antonym dictionary, and meaning specificity) ways to create a model, as shown in Table 6.

Due to practical concerns, I did not consider models involving scope detection via a parse tree and scope resolution via an antonym dictionary. This is due to the expensive operation of extracting words from nodes for the implementation I adopted (one model falls under this category). While information about the sentiment of each node is easy to extract, negation resolution via an antonym dictionary requires individual word tokens which are buried deep down in the tree at the leaf nodes. I did not implement changes to the structure of parse trees that would allow cheap accessing of word tokens. Future research should consider combining parse trees and an antonym dictionary.

---

in fact the scope identified by the incorrect parse tree generated by the Stanford parser.

7.3 Sentiment aggregation

Once negation has been resolved, the sentiment-bearing terms must be aggregated to arrive at a single score for the sentence. I considered two methods to sentiment score aggregation at the sentence level: *flat*-average aggregation and *structure*-based aggregation. To obtain the overall review sentiment, I took the average of the sentiment scores across all sentences.

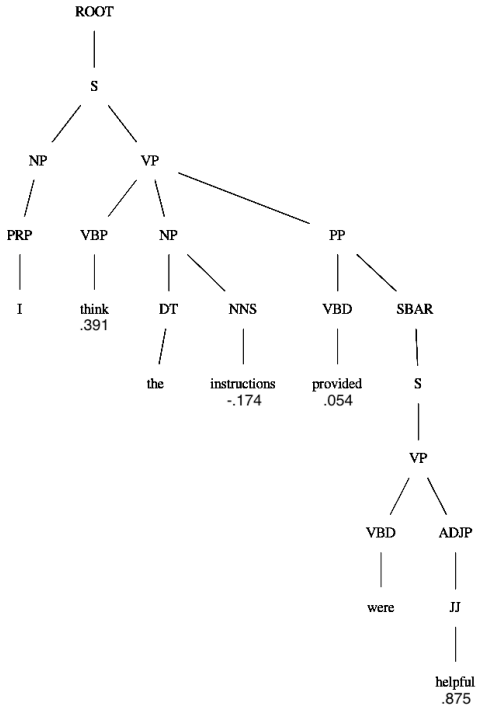

Figure 4: Example of a structure parse tree with labeled sentiment scores.

The flat-average aggregation approach simply averages all available sentiment values; this is equivalent to $\frac{1}{N} \Sigma\ s_i$, where $\Sigma\ s_i$ is the sum of all sentiment scores and N is the number of sentiment-carrying terms. In contrast, the structure-based aggregation approach relies on the form of the parse tree. An average is taken at each layer of the tree, starting with the leaves and working recursively up to the root node. More specifically, for each node with children, that node's sentiment value is the average of its child sentiment values (i.e., the sentiment score of the parent = $\frac{1}{C}\ \Sigma s_c$ where C is the number of children).

I demonstrate the differences in these two approaches in Figure 4, where I consider the sentence "I think the instructions provided were helpful." There are several sentiment-bearing terms: "think"=0.391, "instructions"=-0.174, "provided"=0.054, and "helpful"=0.875. A flat-average aggregation approach would yield a final sentiment score of 0.287, taking an average of words that carry sentiment values (Sentiment=$\frac{1}{4}(0.391 + (-0.174) + 0.054 + 0.875) = 0.287$). A structure-based aggregation approach would yield a different sentiment score: first, the VP "were helpful" inherits the sentiment score of 0.875 from "helpful"; then, the PP "provided were helpful" arrives at an averaged sentiment score of .465; the NP "the instructions" is assigned a sentiment score of -0.174 while the VBP

"think" carries sentiment score of .391. The sentiment bearing terms are averaged at node VP "think the instructions provided were helpful" with a sentiment score of .227 and finally propagate back to the root where the whole sentence "I think the instructions provided were helpful" is assigned of a sentiment score of .227.

8 Evaluation of negation handling approaches: Results of the sentiment analysis pipeline

8.1 25 negation handling models

Table 6 shows the sentiment analysis models that were included in evaluation. I first included a baseline model that involves no negation handling. The remaining models are the result of a factorial crossing of negation resolution method, negation scope detection, and aggregation method. I considered 3 methods for negation scope detection (4-grams window[7], parse tree, and machine learning based classifier), 5 methods for negation resolution (inverting, asymmetrical shift, antonym dictionary, meaning specificity lookup, and meaning specificity lite), and 2 methods for sentiment aggregation (flat and structure-based). This yielded 3 x 5 x 2 = 30 sentiment analysis models.

However, not all of these were investigated, because I removed combinations with negation scope detection that used a parse tree and aggregation that used a flat-average (five models fall under this category since there are five methods for negation resolution). This is because a parse tree represents more sophisticated linguistic knowledge compared to a flat aggregation. Therefore, if I have already gained access to the information provided by a parse tree, I see no motivation to disregard it and revert to a naïve form of flat aggregation. Therefore, when the parse tree is used for negation scope detection, I only considered the structural aggregation method, fully utilizing the syntactic information available during sentiment aggregation.

The total number of models compared in my current project is 25 (1 baseline model + 30 sentiment analysis models - 5 parse tree negation scope combined with flat aggregation models - 1 parse tree negation scope combined with antonym dictionary and structural aggregation).

[7] I selected 4-grams due to my pilot analysis that compared the accuracy of 1-, 2-, 3- and 4-grams. This analysis used inverting for negation resolution and flat aggregation. The best performing window size was 4, and so I used 4-grams in the main analyses.

Models	Negation Scope Detection	Negation Resolution	Aggregation
1	None	None	Flat
2	4-grams	Inverting	
3		Asymmetrical Shift	Flat
4		Antonym Dict	
5		Meaning Spec	
6		Meaning Spec Lite	
7		Inverting	
8		Asymmetrical Shift	Structure
9		Antonym Dict	
10		Meaning Spec	
11		Meaning Spec Lite	
12	Parse Tree	Inverting	
13		Asymmetrical Shift	Structure
14		Meaning Spec	
15		Meaning Spec Lite	
16	Negtool	Inverting	
17		Asymmetrical Shift	Flat
18		Antonym Dict	
19		Meaning Spec	
20		Meaning Spec Lite	
21		Inverting	
22		Asymmetrical Shift	Structure
23		Antonym Dict	
24		Meaning Spec	
25		Meaning Spec Lite	

Table 6: Models that were evaluated, which combined different approaches to negation scope detection, negation resolution, and sentiment aggregation.

8.2 Test Datasets

I now describe the test datasets used to evaluate the different sentiment analysis models in Table 6. I introduce two test datasets, easy reviews and hard reviews, to evaluate the models. I then present the motivation for using the each dataset as well as the inferences that can be made.

8.2.1 Easy reviews dataset

Since my goal is to test different negation handling strategies, an appropriate test dataset would be reviews that contain many instances of negation. The easy reviews test set was generated by extracting 10,000 reviews with at least one negation word from a larger Amazon product review corpus (He and McAuley, 2016; McAuley et al., 2015). A list of negation words used is included in Table 2. The distribution of test data based on target sentiment is included in Table 7.

The significance of this easy dataset is that the distribution closely approximates the actual usages of negation in human-generated reviews. This is because no additional constraints were placed when I sampled reviews which contained negations. As I pointed out in the introduction, there were multiple interpretations of negation depending on the context; the easy reviews test dataset includes the sentiment-bearing reviews in their respective proportions. Therefore, it is useful because it can reveal the incentive for even applying a negation handling model in a real world application. In particular, I will be looking at the difference in performance between any negation handling model and my baseline model that ignores negation handling. I hypothesized that a model that ignores negation would perform poorly. However, I was unsure of the magnitude of decline in performance since the sentiment of negated words would be averaged with other sentiment-carrying terms in the review. Additionally, I will be evaluating whether any negation handling model performed poorly, which would indicate that the model is not applicable to real world negation distribution.

8.2.2 Hard reviews dataset

The hard reviews test dataset is intended to be a more demanding assessment which can potentially differentiate among all negation handling models. While the easy reviews test dataset may also be tested on my baseline model, the baseline model should not be able to correctly classify any reviews for the hard reviews test dataset. This way, any differences in performance can be attributed to the differences in

negation handling strategies (and their combinations of negation scope detection and aggregation methods).

The hard reviews test dataset is created by including two additional criteria in my sampling of test instances: (1) the baseline model (which ignores negation) predicted the sentiment inaccurately and (2) the target star rating is non-neutral. Criterion 2 was added to only consider cases where where negation maximally changes the overall sentiment of a review (from positive to negative, or negative to positive), filtering out less extreme examples (from positive or negative to neutral). Future research could consider including neutral targets to further differentiate the nuances of negation handling models.

An example review from the hard reviews test dataset read: "This product truly did not live up to the expectations; or advertised results! Will not repurchase. Do not recommend it". This review has a negative truth polarity (star rating of 1). However, a model ignoring negation would label the review as positive based on positive words such as "live," "expectations," "repurchase," and "recommend." The usage of negation completely inverts the sentiment from positive to negative. Without proper negation handing, a model will struggle to label the intended polarity. For example, the baseline model labels this sentence as having a sentiment score of .14 (which is effectively neutral). In other words, because negation has a strong impact on these examples, I should be able to see the impact of more or less-effective strategies for negation handling.

I again sampled 10,000 examples of hard reviews from the Amazon product review corpus (He and McAuley, 2016; McAuley et al., 2015). The distribution of test data based on target sentiment is included in Table 7. Note that while it is beneficial to keep the proportion of sentiment targets for easy reviews to approximate the actual distribution, it might be less desirable to do so for hard reviews if the resulting distribution is unbalanced. In my case, the distribution came out to be severely unbalanced. This complicated the interpretation later on since high performance on the hard reviews test dataset may indicate good negation handling model in general, or a negation handling model that only performs well in predicting negative reviews. The current hard reviews dataset did not allow us to distinguish among the two. Future research should investigate model performance run on a balanced hard reviews test dataset.

8.3 Evaluation metric

To assess the performance of negation handling models, I compare model predictions with the ground truth. Since the test datasets were extracted from Amazon

Table 7: Breakdown of test data based on target sentiment

Test data	**Positive**	**Negative**	**Neutral**
Easy	6945	1965	1090
Hard	601	9421	0

reviews, I have access to the star ratings (1 star - 5 stars) associated with each review; these can be used as user-labeled ground truth of the intended sentiment. I binned the star ratings into three sentiment classes (positive, neutral, and negative) using the mapping in Table 8. Similarly, I converted real number model predictions in the range [-1,1] to sentiment bins with the mapping described in Table 5.

Star ratings	Sentiment bins
$1-2$	Negative
3	Neutral
$4-5$	Positive

Table 8: Mapping from star ratings to sentiment bins

Once I converted both model predictions and their respective targets to the same scale (i.e. sentiment bins of positive, neutral, and negative), model performance can be evaluated by calculating the rand index (**RI**) (Rand, 1971). RI can be understood as calculating the proportion of correct classifications. A correct classification is when the model predicted the same sentiment class as the target. The formula for RI in included in Equation 6.

$$RI = \frac{correct\ classifications}{all\ reviews = 10,000} \tag{6}$$

While this is the ideal metric to be implemented, I find RI sub-optimal in this project as it produces scores that are heavily left-skewed. Most models have performances clustered at the lower end of the scale, and thus no differences between negation handling techniques can be identified. The issue of the skewed distribution did not come from the RI, but instead the complication that rises when predicting sentiment ratings from reviews. I identified one prominent factor that inspired me to introduce a more lenient accuracy scoring metric.

Specifically, I revised my definition of correct classification in light of the "neutralization" effect (Lak and Turetken, 2014). Lak and Turetken (2014) in their investigation of the agreement between text-based opinion and human-generated ratings found that when it comes to expressions of opinion, people tend to adopt a neutral tone. That is, regardless of sentiment rating (positive, neutral, or negative), the set of word choices overlap greatly: in all cases people chose to use neutral words to express their opinions for the most part. As such, Lak and Turetken (2014) observed that most predictions of sentiment analyzers lie in the neutral region. Without sophisticated techniques sensitive to nuances in natural language, it is a significant challenge to encourage non-neutral predictions in models.

I would like our evaluation metric to be more informative about the best negation handling model. Therefore, I see a need to slightly revise my evaluation metric to allow comparisons between different negation handling strategies while compensating for the limitations imposed by the neutralizing effect from sentiment analyzers. I introduce a more lenient accuracy scoring metric where model prediction of neutral sentiment is less wrong than model predictions of opposite sentiment (target being positive and model predicted negative or target being negative and model predicted positive). The new metric, which I call *partial neutral* (**PN**), will still award 1 full point to correct classifications of any class (positive as positive, neutral as neutral, negative as negative); however, it will award partial credit (0.5) to any predictions labeling polarized input (i.e., positive or negative sentiment) as neutral (see Equation 7). So, RI serves as a lower bound for the PN score. In the case where a model simply predicts neutral sentiment for all input, the maximum partial neutral score would be .50.

$$PN = RI + \frac{.5 * [neutral_{model} | target_{positive/negative}]}{all\ reviews = 10,000} \tag{7}$$

While I am aware of the importance of the neutral classification, especially when it comes to sentiment analysis (Koppel and Schler, 2006; Hamed et al., 2016), this research focused on accurately labeling the coarse classification of two extreme sentiment classes: positive and negative sentiment classes. This is because it is more damaging to a classifier to grossly misclassify sentiment of the opposite polarity. For example, it is more harmful to conclude that customers of a certain product are happy (positive sentiment) while they are not (negative sentiment) or misapprehend satisfaction level as low (negative sentiment) while the general consensus is high (positive sentiment). In other words, it raises a greater concern when sentiment analyzers predict opposite sentiment classes than labeling them as

neutral. Therefore, I think it is logical to assign partial credit to false labeling of neutral sentiment. Results of evaluation using partial neutral metric are informative for cases where the consequence of labeling a polarized input as neutral is not as expensive as labeling the sentiment the polar opposite, such as considering the consensus of public on a product.

8.4 Results

A complete version of the results can be found in Table 9. Given that the main focus of this research is to find the best negation resolution method, I calculated the mean performances across different negation resolution methods, collapsing negation scope detection and aggregation methods. A visualization of the analysis is included in Figure 5.

For the easy reviews test dataset, the best performing model the one using the 4-grams negation scope detector, inverting negation resolution, and the structural aggregation method (partial neutral score of .6376). However, all five negation resolution methods performed equally well, with performances falling in the range [.56 - .64]. One surprising observation was that performance of the baseline model which ignored negation was high (ranked 9th out of 25 models). Figure 6 provides a visual representation for the distribution of performances of each of the 25 models, along with the baseline model which was represented by the red boundary line.

The hard reviews test dataset provided a better window for differentiating between the negation resolution methods. The best performing model was the one using the Negtool negation scope detector, meaning specificity lite negation resolution, and the flat aggregation method (partial neutral score = .6575). The best negation resolution method was meaning specificity lite with mean partial neutral score of .604 (range [.58 - .66]). Notably, all models of meaning specificity lite ranked higher than any model from the second best performing negation resolution method, inverting, which had a mean partial neutral score of .519 (range [.50 - .56]) (see to Figure 7.

Since negation resolution is part of a bigger sentiment analysis pipeline, I performed the same aggregate analysis, calculating the mean performances of models, on negation scope detectors (4-grams, Negtool, and parse tree) and aggregation methods (flat and structural) to gain a deeper understanding on how each stage of the analysis contribute to the final performance score. Note that I did not include performance of the baseline model when I collapse model performances across a particular negation scope detection or aggregation method because the baseline model had a partial neutral score of 0 on the hard reviews test dataset. From my

Table 9: Summary of Results. Bolded values represent the best performing model within the specific test dataset.

Models / Datasets	**Partial Neutral**			
	Easy		Hard	
	Flat	Structural	Flat	Structural
Baseline (None)	.6287	-	0.0	-
4-grams				
Inverting	.6116	**.6376**	.4986	.4968
Asymmetrical shift	.6070	.6330	.4435	.4357
Antonym dictionary	.6147	.6321	.2728	.2823
Meaning specificity	.6067	.6335	.4421	.4325
Meaning specificity Lite	.5873	.6067	.5788	.5693
Parse Tree				
Inverting	-	.6348	-	.5012
Asymmetrical shift	-	.6299	-	.4405
Meaning specificity	-	.6035	-	.5757
Meaning specificity Lite	-	.6013	-	.5827
Negtool				
Inverting	.6066	.6319	.5573	.5416
Asymmetrical shift	.6004	.6263	.4799	.4667
Antonym dictionary	.6098	.6328	.3393	.3356
Meaning specificity	.6013	.6270	.4772	.4621
Meaning specificity Lite	.5572	.5877	**.6575**	.6355

Note:

Bolded values represent the best performing model for the dataset.

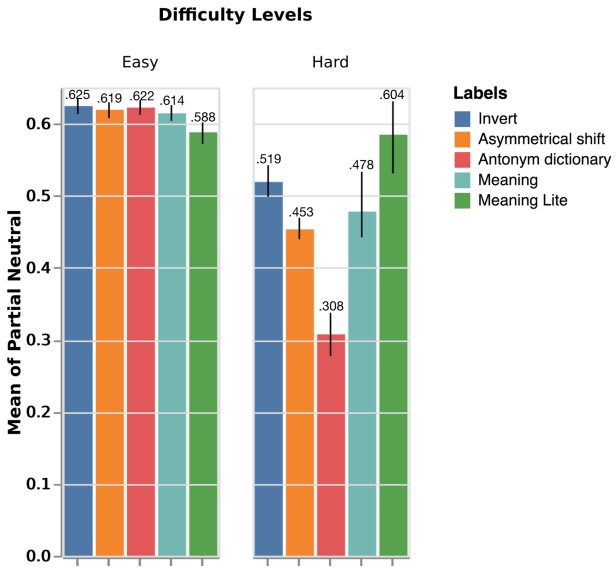

Figure 5: Visualization of negation resolution, collapsed across scope detection and aggregation. Error bars shows 95% confidence interval for the mean.

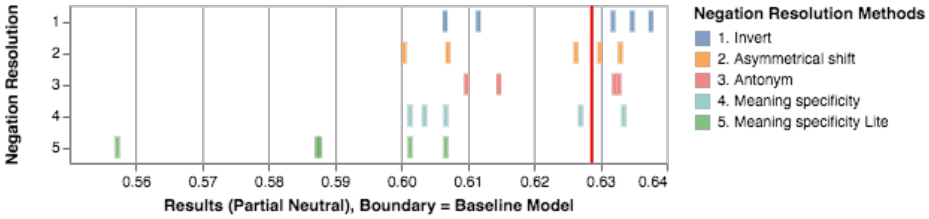

Figure 6: Visualization of performances of 25 easy review models. The red boundary line represents the partial neutral score achieved by baseline model.

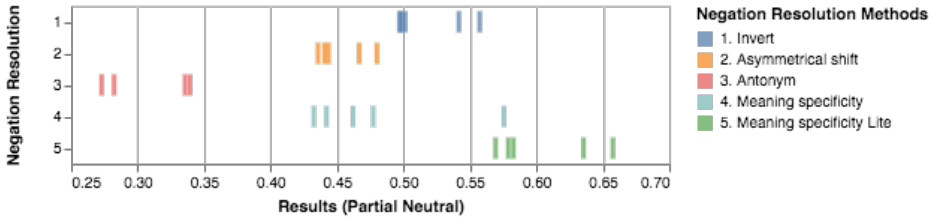

Figure 7: Visualization of performances of 25 hard review models. The baseline model had a partial neutral score of 0 and therefore is omitted from the figure.

analysis (see Figure 8), there was no specific incentive for utilizing a particular negation scope detector or aggregation method for both easy and hard reviews. Although it seems that choosing a parse tree negation scope detector yielded better results than using 4-grams, a Bayesian ANOVA revealed that it is 3.74 times more likely to occur under a model without including an effect of negation scope detectors. That is, the analysis was in favor of the null hypothesis: there are no difference between the mean partial neutral scores of different negation scope detectors.

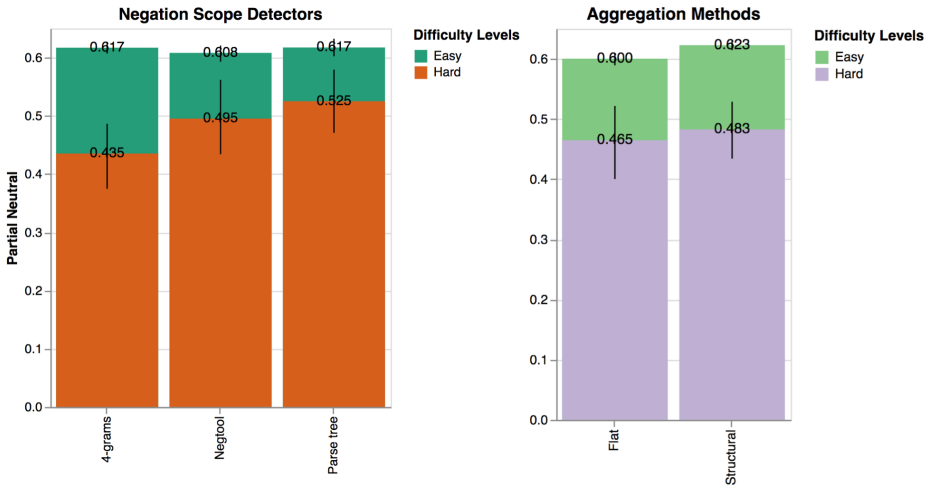

Figure 8: Left: Visualization of negation scope detectors, collapsed across negation resolution and aggregation methods. Right: Visualization of aggregation methods, collapsed across negation scope detectors and negation resolution. Both: Error bars shows 95% confidence interval for the mean.

9 General discussion

Here I have investigated how incorporating meaning specificity could benefit negation handling approaches in sentiment analysis. In particular, I have provided an empirical comparison of different negation handling strategies implemented in a sentiment analysis pipeline. I compared meaning specificity approaches (lookup and lite) against three previous strategies: inverting, asymmetrical shift and antonym dictionary. The meaning specificity approach I introduced uses frequency

and inverse dispersion to approximate meaning specificity, harnessing intuitions about factors that seem to impact a word's meaning specificity.

In this section, I elaborate on the results obtained from running two datasets of varying difficulties, highlighting cases where meaning specificity negation resolution outperformed other methods. I discuss the factors contributing to good performance and similarly investigate causes where applying meaning specificity is less beneficial. Finally, I conclude by discussing the limitations of current work as well as presenting suggestions for future work.

9.1 Inferences to be drawn from datasets of varying difficulty

To study the effects of different negation resolution strategies, it is important to have datasets that would reveal where the strategies differ. Notably, I found that the "easy dataset", containing reviews that had least one negation word didn't do this. The different strategies had very similar performance. In contrast, the "hard reviews" dataset revealed different performance for different negation resolution strategies. In this subsection, I present the inferences to be drawn from each dataset based on observations of the results, and discuss their contributions in my search for the best negation resolution method.

9.1.1 Easy Reviews Dataset

My results showed that for easy reviews, all negation handling methods performed similarly and obtained comparable results, with partial neutral scores in the range of [.56 -.64]. Moreover, the baseline model, which ignores negation, ranked high in the list, 9th out of 25 (partial neutral score of .629 compared to the best performing model of .638). It seems counter-intuitive that doing nothing is better than doing something, even though negation is occurring. I investigate this observation from two directions: (1) the reason for the high performance of the baseline model, and (2) reasons for negation handling models to perform worse than the baseline model.

One plausible explanation for the high performance of the baseline model is that negation is used in conjunction with other words that have strong sentiment cues, making the effect of negation on the overall sentiment less pronounced. For instance, consider this sample review excerpt: "This case is as cute as it is durable. Your phone sits in a rubber casing that fits very snug. Your phone won't be falling out." The review excerpt expressed positive sentiment with cue words such as "cute," "durable," "fits," and "snug." These positive sentiment terms collectively overpower the negative sentiment of "falling." That is, doing nothing to negate the

negative sentiment of "falling" doesn't hurt much – it's overpowered by the positive sentiment terms already. So, the resulting sentiment of this specific review with or without the negation phrase is fairly similar (all models including the baseline model predicted positive sentiment, specifically 4 stars, with sentiment values around .2). This highlights that the contribution of the negation in these review sentences is small compared to the sentiment already present.

On the other side, models performing some kind of negation handling performed worse than the baseline model may be explained by the models failing to arrive at the intended interpretation of negation. For instance, the review "It is and does exactly what the description said it would be and would do. Couldn't be happier with it" would be classified positive by a baseline model ignoring negation; however, it might be classified as neutral by negation-handling models because the phrase "couldn't be happier" does not diminish the intensity of happiness but rather intensifies the expression. In other words, "couldn't be happier" is interpreted as less positive than "happier" alone. Considering that there might be other examples of this kind, models performing negation handling incorrectly adjusted their predictions toward the opposite sentiment, thereby arriving at performance scores slightly lower than the baseline model.

Overall, the easy reviews test dataset is valuable in that it captured common usages of negation. Results from the easy dataset serves as a sanity check as any negation resolution method should be broadly applicable to different negation cases, including easy ones. When coupled with results from the hard reviews dataset, the negation resolution method with the least variance in performance across these two datasets can be considered the most robust technique since it would indicate that the method performs well irrespective to review difficulty.

9.1.2 Hard Reviews Dataset

While all models performed comparably well on the easy reviews test dataset, the hard reviews test dataset was designed to be a more rigorous test to differentiate among the models. Specifically, the hard reviews test dataset is a compilation of reviews where the baseline model that ignored negation failed to predict the target sentiment accurately. In other words, hard reviews are instances where negation really mattered and ignoring negation leads to blatant errors. From the distribution of model performances, the hard dataset successfully distinguished different negation handling strategies, with model performance in the much wider range of .27-.66. Therefore, hard reviews allowed us to infer the method that handles negation more robustly.

With hard reviews, the top performing models were ones that utilize meaning specificity as their negation resolution method, with the most obvious improvement observed when applying the lite version of meaning specificity. Specifically, the top six models had meaning specificity as their negation resolution methods, with the model combination of negtool scope detector + lite version of meaning specificity negation resolution + flat having the highest partial neutral score of .6575. Results from the hard reviews dataset illustrate that applying the meaning specificity approach is more rewarding when negation plays an important role in influencing the final target sentiment category.

9.2 Negation resolution strategy highlight: Meaning specificity lite as a robust general negation handling strategy

9.2.1 Assessing the effort and reward for applying meaning specificity lite

Results from the hard reviews particularly highlighted the value of meaning-specificity-based approaches to negation resolution: the top 5 models were models utilizing the meaning specificity lite approach. This suggests that it is beneficial to include components of meaning specificity when handling hard instances of negation in sentiment analysis, where negation completely changes the polarity of an expression.

However, the same isn't true for the easy reviews. Although all models performed similarly on the dataset, meaning specificity lite ranked last compared to other negation handling methods, with an averaged partial neutral score of .588, compared to the best performing .625 of inverting. Given this, if it is known beforehand that the negation instances are rarely hard in the sense I defined here, it's better to implement the inverting technique of negation resolution which requires less effort to implement.

On the other hand, when the nature of the test dataset is unknown (as may often be the case), it seems useful to apply the meaning specificity lite approach to negation resolution. Meaning specificity lite is the most robust of all the techniques I investigated, as evidenced by the small variance in partial neutral scores for both easy and hard reviews (with .588 for easy and .604 for hard). Therefore, this method is capable of handling easy reviews, which are found more abundantly in real world data sets; it's additionally more reliable for dealing with hard reviews, making fewer mistakes that are really bad, such as classifying the intended sentiment as the opposite.

9.2.2 Meaning specificity lite being a less accurate variant of meaning specificity

Recall that meaning specificity lite was intended as a lightweight version of the original proposed meaning specificity negation resolution method. Instead of a lookup table mapping sentiment tokens to meaning specificity components, it relied on distributions from which meaning specificity components (i.e., frequency and inverse dispersion) are sampled from. Interestingly, these models performed better than those that utilized meaning specificity lookup, especially for the hard reviews dataset.

9.2.3 Incomplete adherence to imperfect underlying assumptions of meaning specificity afforded better performance

It might seem counterintuitive that utilizing a less precise meaning specificity method, that is, random sampling from distributions rather than using the actual value computed from a corpus sample, boosted performance. However, note that the meaning specificity lite approach did not sample meaning specificity components from completely random distributions. Rather, the method injected randomness in a structured way, based on similar sentiment scores: the meaning specificity components needed to calculate the negated sentiment score were based on the affirmative sentiment bins they fell into. This structured randomness somehow moved the performance in a more desirable direction.

Similar findings have been noted in the realm of language development modeling; there are cases where better developmental performance occurrs when the learning model had sub-optimal inference (Phillips and Pearl, 2015). More specifically, modeled learners who had to approximate inference for the specified computation performed better than modeled learners who accomplished this inference optimally. Phillips and Pearl (2015) suggested that this was because the computation that the learner was trying to accomplish was itself sub-optimal – so optimal inference for that sub-optimal computation yielded an outcome that was itself sub-optimal. In contrast, approximate inference pushed the modeled learner away from the optimal outcome of that sub-optimal computation, and for whatever reason, towards a better outcome.

Here, frequency and inverse dispersion are heuristics used to approximate meaning specificity. In this way, they can be considered a sub-optimal computation of meaning specificity. In addition to that, the equation used a likely sub-optimal set of weights to combine the features, as these were estimated from a small gold

standard set. Using the same reasoning as Phillips and Pearl (2015), perhaps approximating these components pushes the model towards a better outcome, because it's no longer adhering so strictly to this sub-optimal computation of meaning specificity. As with the results of Phillips and Pearl (2015), further research is needed to pinpoint why this way of approximating meaning specificity improved performance.

9.3 Limitations and Future Work

The current work served as an exploratory analysis of the contributions of meaning specificity in negation handling, as applied in sentiment analysis. With this in mind, I consider some limitations of the current study, along with actionable future directions involving how to better evaluate negation handling models and how to more fruitfully incorporate meaning specificity into a negation handling strategy.

9.3.1 Evaluation

In the current project, two datasets were used to empirically compare different negation resolution strategies: easy and hard reviews. As an exploratory analysis, both test datasets were extracted without balancing, i.e., the test subset I extracted most likely resembles the actual distribution in the original amazon review dataset (He and McAuley, 2016; McAuley et al., 2015). While keeping the proportions of positive, neutral, and negative reviews close to their respective true distribution is useful for easy reviews, the current research neglected the importance of balancing the hard reviews test dataset. Recall that the hard reviews test dataset was populated with negative reviews at a proportion of .94. As such, it is unclear if a high performing model on my hard reviews test dataset would continue to perform well if tested on positive reviews. Given this, it would be useful to evaluate the meaning specificity strategies I investigated here on a balanced dataset of positive and negative hard reviews.

Nonetheless, regardless of the outcome (i.e., whether meaning specificity negation resolution method performs well on a balanced test dataset), the results here are still interesting. The ideal case would be that meaning specificity models performed equally well on balanced datasets, suggesting the approach I took is not dependent on the valence of the review. If instead the meaning specificity approaches don't perform better on balanced datasets, this suggests the approach is most useful on negative valence reviews. This would be quite interesting because in general, sentiment analyzers perform worse on negative valence reviews (Dhaoui

et al., 2017). So, in this case, meaning specificity would offer a targeted away to improve performance on negative reviews.

Another limitation with the current evaluation concerns the integrity of the target. Recall that in the current research I regarded the user-provided star rating as the target or ground truth which I compared my model predictions against. However, several studies have pointed out the tendency for reviewers to write and rank differently, termed the *text-rating inconsistency* (**TRI**) phenomenon (Lak and Turetken, 2014; Geierhos et al., 2015). In the current study, I treated star ratings of reviews as ground truth when evaluating models; this implicitly assumes consistency in text rating. Nevertheless, given what I have access to, I regard star ratings as a good enough window into the coarse sentiment of a text-based review (Lak and Turetken, 2014).

9.3.2 Elements of meaning specificity and ways of combining them

The most profound component of a lexicon-based sentiment analyzer is its sentiment dictionary. In general, having an accurate and comprehensive sentiment lookup is the foundation for further applications. However, it's often difficult to create such a lookup table. In the current study, my compiled sentiment dictionaries consist of manual human-labeled lexica mixed with automatically-generated ones; this led to the dictionary prioritizing comprehensiveness over accuracy. In addition to this, the current project did not consider aspects of word senses or contexts. For instance, "thick", when used in the context of a winter jacket or in the context of a phone, has a different sentiment score. Future research could consider utilizing a better sentiment dictionary, with sentiment manually annotated by humans via crowd-sourcing, and hopefully obtain a better sentiment dictionary, which would be the foundation of any sentiment analyzer.

Another open avenue of future work concerns the specific meaning specificity approach I pursued in the current study. While it was the most robust strategy compared to other approaches, there are several aspects where my implementation could be improved. These include: (1) elements involved in estimating meaning specificity, (2) the gold standard or training data used to tune the negated sentiment equation, and (3) methods of combining the elements. I discuss each in turn.

First, I consider the elements I used to heuristically approximate meaning specificity: frequency and inverse dispersion. My results showed that models utilizing meaning specificity lite, which sampled from distributions of frequency and inverse dispersion respectively, were robust across both the easy and hard reviews test dataset. Nonetheless, the exact relationship between those two components and

meaning specificity is yet to be determined. One potential solution is to conduct a behavioral study where users rate the specificity of different words. Correlation or dependence analysis would reveal any statistical relationship between my proposed meaning specificity elements (frequency and inverse dispersion) and meaning specificity.

Another aspect that is worth investigating is a more accurate and indicative gold standard. Since my negated sentiment equation is tuned by performing multiple regression analysis on the gold standard, future studies should experiment with a more balanced and larger corpus for a gold standard. In particular, the current gold standard consists of 42 entries (25 positive, 5 negative, and 12 neutral affirmative entries mapped to 4 negative, 9 neutral, and 29 positive negated sentiment); these entries were extracted from an automatically-generated lexicon (Kiritchenko et al., 2014). Therefore, an equation tuned to a larger, more balanced, and possibly more accurate gold standard is likely to improve performance. Although I manually inspected the gold standard to catch entries that didn't align with my intuitions, an attainable ideal would be to construct the gold standard from human judgments on the negated sentiment.

Another natural question concerns the way to combine the elements that relate to meaning specificity, and then are used to calculate the negated sentiment. Currently, I utilized multiple regression to determine the weights of each component in a linear equation. However, the coefficients for variables in the equation could be determined by other ways such as a support vector machine (SVM). On top of that, the elements may also be combined in non-linear ways using techniques such as a multilayer perceptron or decision tree. Future research can investigate the best method for combining meaning specificity elements to generate a negated sentiment score.

Lastly, a surprisingly finding in the current study was that the meaning specificity lite negation handling strategy was the most robust technique, obtaining results with lowest variance across reviews of different difficulty levels. As noted in section 9.2.3, it's unclear why this occurred. Future research can ascertain if this continues to occur across different meaning specificity implementations, and if so, what the underlying cause might be.

10 Conclusion

This project evaluated meaning specificity as a potential feature to be included in negation handling of sentiment analysis. Meaning specificity was quantified by

frequency and inverse dispersion, as well as the interaction between the two. It was validated that there is information to be gained demonstrated by reduction in entropy, taking meaning specificity as an additional feature on top of sentiment scores. By incorporating a meaning-specificity-based negation handling approach into a sentiment analysis pipeline, an improvement in performance was achieved when tested on a harder subset of negation instances. Performance of a particular meaning specificity negation approach (meaning specificity lite) was also most stable across the two test sets with variable difficulty, as evidenced by its lowest variance in performance across the easy and hard reviews test datasets. Therefore, I find positive support for incorporating meaning specificity in negation handling for sentiment analysis.

Acknowledgement

I would like to thank Alan Yuen and Lisa Pearl for their thoughtful comments and suggestions.

Appendix A Gold Standard with Affirmative and Negated Sentiment Scores

Table 10: Forty-two adjectives and adverbs I regarded as the gold standard, and used to tune equations for asymmetrical shift and meaning specificity negation resolution approaches.

Word	**POS**	**Affirm**	**NRC Affirm**	**NRC Negated**	**Negation Freq**
able	Adjective	0.25	0.28	-0.50	278,625
alone	Adverb	-0.55	-0.03	0.26	38,372
attractive	Adjective	0.68	0.30	-1.51	10,756
bad	Adjective	-0.50	-0.92	0.19	471,908
best	Adjective	0.81	1.12	-0.02	190,440
clear	Adjective	0.25	0.37	-1.15	86,683
disappointed	Adjective	-0.58	-1.96	1.53	253,215
due	Adjective	0.04	0.20	-0.34	21,083
far	Adverb	0.02	0.10	0.01	40,576
free	Adjective	0.56	0.36	-1.31	24,707
full	Adjective	0.48	0.35	-0.44	42,207
further	Adverb	-0.04	-0.46	0.82	65,724
good	Adjective	0.66	0.74	-1.08	643,979
great	Adjective	0.73	1.42	-0.34	270,430
happy	Adjective	0.91	1.07	-1.49	171,241
high	Adjective	0.02	0.20	-0.80	62,514
important	Adjective	0.33	0.04	-0.53	23,059
impressed	Adjective	0.81	1.18	-1.51	78,127
impressive	Adjective	0.60	1.04	-1.04	10,202
interested	Adjective	0.48	0.02	-1.12	49,546
live	Adjective	0.11	0.46	-0.52	42,389
necessary	Adjective	-0.09	-0.24	-0.01	55,632
new	Adjective	0.41	0.66	-0.89	79,927
nice	Adjective	0.72	0.91	-1.90	32,916
often	Adverb	-0.02	-0.13	0.35	40,983
open	Adjective	0.39	0.40	-0.62	35,215
possible	Adjective	0.10	0.03	-0.55	43,309
prepared	Adjective	0.13	0.02	-0.34	12,864

pretty	Adverb	0.77	0.17	-0.72	24,281
ready	Adjective	0.52	0.57	-0.97	36,285
safe	Adjective	0.56	0.24	-1.50	20,885
satisfied	Adjective	0.53	1.09	-0.69	32,789
simply	Adverb	0.11	1.04	-0.83	26,697
special	Adjective	0.74	1.06	-0.90	23,990
strong	Adjective	0.08	0.62	-0.42	80,894
together	Adverb	0.03	0.26	-0.10	23,836
truly	Adverb	0.28	0.46	-0.10	13,363
well	Adverb	0.47	0.35	-1.20	232,925
willing	Adjective	0.22	0.06	-1.04	23,269
worried	Adjective	-0.23	-0.69	-0.08	21,753
wrong	Adjective	-0.44	-1.01	-0.09	29,418
young	Adjective	0.20	0.21	-0.90	13,571

References

Sasha Blair-Goldensohn, Kerry Hannan, Ryan McDonald, Tyler Neylon, George A Reis, and Jeff Reynar. Building a sentiment summarizer for local service reviews. In *WWW workshop on NLP in the information explosion era*, volume 14, pages 339–348, 2008.

Jorge Carrillo-de Albornoz and Laura Plaza. An emotion-based model of negation, intensifiers, and modality for polarity and intensity classification. *Journal of the Association for Information Science and Technology*, 64(8):1618–1633, 2013.

Chedia Dhaoui, Cynthia M Webster, and Lay Peng Tan. Social media sentiment analysis: lexicon versus machine learning. *Journal of Consumer Marketing*, 34 (6):480–488, 2017.

Martine Enger, Erik Velldal, and Lilja Øvrelid. An open-source tool for negation detection: a maximum-margin approach. *SemBEaR 2017*, page 64, 2017.

Michaela Geierhos, Frederik Simon Bäumer, Sabine Schulze, and Valentina Stuß. " i grade what i get but write what i think." inconsistency analysis in patients' reviews. In *ECIS*, 2015.

Stefan Th Gries. Dispersions and adjusted frequencies in corpora. *International journal of corpus linguistics*, 13(4):403–437, 2008.

AL-Rubaiee Hamed, Renxi Qiu, and Dayou Li. The importance of neutral class in sentiment analysis of arabic tweets. *Int. J. Comput. Sci. Inform. Technol*, 8: 17–31, 2016.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.

Svetlana Kiritchenko and Saif Mohammad. The effect of negators, modals, and degree adverbs on sentiment composition. In *WASSA@ NAACL-HLT*, pages 43–52, 2016a.

Svetlana Kiritchenko and Saif M. Mohammad. Sentiment composition of words with opposing polarities. In *Proceedings of The 15th Annual Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL), San Diego, California, 2016b.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics, 2003.

Moshe Koppel and Jonathan Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109, 2006.

Parisa Lak and Ozgur Turetken. Star ratings versus sentiment analysis–a comparison of explicit and implicit measures of opinions. In *System Sciences (HICSS), 2014 47th Hawaii International Conference on*, pages 796–805. IEEE, 2014.

Wei-Yin Loh. Fifty years of classification and regression trees. *International Statistical Review*, 82(3):329–348, 2014.

Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

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

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*, 2013.

Roser Morante and Walter Daelemans. Conandoyle-neg: Annotation of negation in conan doyle stories. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, Istanbul*. Citeseer, 2012.

Lawrence Phillips and Lisa Pearl. The utility of cognitive plausibility in language acquisition modeling: Evidence from word segmentation. *Cognitive science*, 39 (8):1824–1854, 2015.

William M Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

Johan Reitan, Jørgen Faret, Björn Gambäck, and Lars Bungum. Negation scope detection for twitter sentiment analysis. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 99–108, 2015.

Nicolas Ruytenbeek, Steven Verheyen, and Benjamin Spector. Asymmetric inference towards the antonym: Experiments into the polarity and morphology of negated adjectives. *Glossa: a journal of general linguistics*, 2(1), 2017.

Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, page 1642, 2013.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37 (2):267–307, 2011.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163–173, 2012.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif Mohammad. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)*, pages 443–447, 2014.