1 Problem Overview

In this brief chapter, we give an overview of how we formulated the problem of recovering the "most likely" total ordering of adjectives. However most of the interesting details are missing, and this chapter serves as a motivation for the rest of the thesis. Suppose we have a cluster of n items, $\mathcal{C} = \{s_1, \ldots, s_n\}$, then we can form the set of all permuations of $\mathcal{C}$ by:

$$\mathbf{\Omega} = \{\omega \in \Pi(\mathcal{C}) : \Pi : \{1, \ldots, n\} \to \{1, \ldots, n\}\}.$$

Now we need to place a distribution over $\mathbf{\Omega}$ so that:

$$\begin{aligned}
\boldsymbol{Pr}[\omega] &= \boldsymbol{Pr}[s_1 < \ldots < s_n] \\
&= \boldsymbol{Pr}[s_1 < s_2 \text{ and } \ldots \text{ and } s_1 < s_n \text{ and } s_2 < s_3 \ldots s_2 < s_n \text{ and } \ldots s_{n-1} < s_n] \\
&= \prod_{i \in \{1,\ldots,n\}, i<j} \boldsymbol{Pr}[s_i < s_j],
\end{aligned}$$

where the last statement follows from the independence assumption among pairwise comparisons. If we assume this $\boldsymbol{Pr}$ exists and can be estimated, then we can pick the most likely ordering ω^* with:

$$\arg\max_{\omega \in \mathbf{\Omega}} \boldsymbol{Pr}[\omega].$$

Some remarks are in order.

Remark 1.1. The independence assumption is a fair characterization of how the data is generated: we surmise when people decide if one word is stronger than another in everyday speech, they are not imagining how they assigned strength to other words in the cluster the last time they spoke of them. Note however the test set may be generated in a very different manner. If the comparisons are done pairwise by turkers, then this approximates the setting of every day speech where

the decisions are independent. However if the annotator is given all n words at once, and asked to rank them, then there is strong sequential dependence between the pairwise comparisons. That is to say if the turker decides that "good" is less intense than "great", then this will inform how he/she places the word "better". Furthermore, the sequence in which the turker makes decisions may lead to different orderings, since the conditional probability that $\boldsymbol{Pr}[s_i < s_j | si < s_k]$ may not be the same as $\boldsymbol{Pr}[s_i < s_j | s_i < s_l]$ for $k \neq l$. However, we do not observe this sequence of decisions and therefore cannot decide if our estimate is close to the true distribution. Thus the independence assumption is, in some sense, the most conservative approach.

Remark 1.2. The complexity of a naive computation of the $\arg\max$ operation is $n^2 n!$ in the size of the cluster $\mathcal{C}$. In theory this operation is prohibitively expensive and there is literature tacking just this problem. In practice our clusters are no more than 5 items large, so it is very manageable even on my two year old laptop.

Remark 1.3. Another possible formulation for a distribution over $\mathbf{\Omega}$ is to assume that there exists a distribution $\mathcal{D}$ over $\mathbf{\Omega}$, and nature (or man) selects ω according to $\mathcal{D}$, and then reveal some pair from ω according to another distribution $\mathcal{D}'$ over the set of all true comparisons implied by ω: $\{s_i < s_j : i, j \in \{1, \ldots, n\}, i < j\}\}$. Again, we cannot measure either of these distributions from the data we have, so this formulation is curious but not helpful.

Remark 1.4. Some readers may find the index notation confusing, so let us be very clear on what we are enumerating in Π and $\prod$. Given vertices s, t, and r, there are $3!$ elements in $\mathbf{\Omega}$ enumerated by Π:

$$s < t < r$$

$$s < r < t$$

$$t < s < r$$

$$t < r < s$$

$$r < s < t$$

$$r < t < s.$$

These are the set of all possible orderings of the three vertices. Next for each ω in $\mathbf{\Omega}$, the indices of $\prod$ enumerates all possible consistent orderings implied by this ω. For example, if $\omega := s < t < r$, then we have $O(n^2)$ comparisons:

$$s < t$$

$$s < r$$

$$t < r.$$

Note a contradictory possibility such as $s > t, s > r, t < r$ is not enumerated, so it could never be picked. Unlike Mohit, we resolve contradiction by not considering it. Finally, since we assume the decisions above are independent, the probability of this ω is:

$$\Pr[s < t < r] = \boldsymbol{Pr}[s < t] \cdot \boldsymbol{Pr}[s < r] \cdot \boldsymbol{Pr}[t < r].$$

In the next few chapters, we offer some ways of determining what the probability should be over, and how to estimate $\boldsymbol{Pr}[s_i < s_j]$ on different data sets. Unless it is explicitly noted, we recover the total ordering from pairwise probabilities using the $argmax$ expression above.

2 Naive Baselines

2.1 "Random" Baseline

In this chapter we present a two simple baselines to estimate $\boldsymbol{Pr}[s < t]$. If we assume the sign $<$ is a Bernoulli variable so that either $s < t$ or $t < s$, then we can assign a uniform probability of $\frac{1}{2}$ to each outcome. Thus all $\omega \in \mathbf{\Omega}$ will have equal probability. This presents an interesting problem to pick the maximum value, since in practice $\mathbf{\Omega}$ is represented by a list, Python will pick the maximum over a list of equal valued items by selecting the first element in the list, therefore the ordering of this list is of profound importance. We could default to randomization, but this introduces some uncertainty in our baseline; picking deterministically is a must to prevent pollution from a bad seed. If we sort $\mathbf{\Omega}$ *lexicographically*, then there many not be any bias given some arbitrary cluster of words (speaking loosely). But if there is any relationship between intensity of words and their surface form, then this sorting will introduce a huge bias. This exactly the case in any cluster with base-comparative-superlative words, sorting alone guarantees 85% pairwise accuracy on the set with only base-comparative-superlative clusters. The solution is to sort and then *reverse* the list, which will create a equally strong bias against the base-comparative-superlative pairs. But this will ensure any gains we have in accuracy comes from how well we incorporate information, not sorting. In conclusion, the baseline is still contaminated by a bias, but it is the "worst possible contamination." We hope the reader finds this argument convincing. Finally, we use the same sort and max function in all future methods so the bias will be consistent.

We test this baseline on three data sets: the original data set annotated by Mohit, the set we constructed using mechanical turks, and finally a set of base comparative superlative adjectives found in the PPDB data base. All results are presented in table 1. Readers who are satisfied with how we constructed the baseline may skip

	N-gram		
Test set	**Pairwise**	**Avg.** τ	**Avg.** $\lvert\tau\rvert$
Mohit	43.0%	-0.04	0.42
Turk	42.0%	-0.07	0.62
BCS	16.0%	-0.69	0.99

Table 1: Random baseline. Note how poorly "randomness" performed on the base-comparative-superlative baseline simply because the lists are sorted lexicographically, while the other sets perform close to random as expected. A high absolute τ is concerning because this suggests that enough of the cluster in the test sets are of length 2 so as to make absolute τ look deceptively high.

the remarks, those who are unsatisfied may skim and critique.

Remark 2.1. Readers who are committed to a truly random baseline have to contend with the fact that certain methods only have a slight edge relative to others, it is important the measures reflects this edge. Since many clusters are only two or three items long, different randomized outcomes could result in a τ of 1.0, 0.333, or -1.0. Therefore randomization will overwhelm any gains in method quality over different runs of the method. In this sense, a deterministic lexicographical ordering presents the best solution to make different methods comparable during development.

2.2 Pointwise Estimation

The next simplest baseline to estimate $\boldsymbol{Pr}[s < t]$ in the spirit of pointwise estimation. Similar to the baseline above, we have two possible events: $\boldsymbol{\Omega} = \{s < t, s > t\}$, and we observe a sequence of comparisons between s and t: $\boldsymbol{S} = \{s < t, s < t, \ldots, s > t \ldots\}$, we can ask what is the probability that the next element we will observe is $s < t$. This is a Bernoulli distribution with parameter p and it is well known that the most likely p is simply:

$$\boldsymbol{Pr}_{\boldsymbol{\Omega}}[s < t] = \frac{|\{s < t \in \boldsymbol{S}\}|}{|\boldsymbol{S}|}.$$

Because this is a baseline, if $\boldsymbol{S}$ is empty then we default to $\boldsymbol{Pr}[s < t] = \frac{1}{2}$, that is to say "we don't know."

2.3 Pointwise Estimation Results

In this section we analyze the results for where the pointwise estimation baseline fails. The goal of the pointwise baseline is to two fold: (1) assess how much information can be gained by considering pairwise comparisons alone, and (2) understand how the three data sets differ. We estimate the probabilities over three data sets, the N-gram data set composed of all occurrences of $s\mathcal{P}t$ for specified patterns $\mathcal{P}$, the PPDB set of paraphrases, and finally a naive combination of the two sets where the data is simply "thrown together." All results are presented below in table 1. Readers who do not like to "lay in bed with the data" may skip to the conclusion, those who are not too smart to count may read each paragraph in detail.

We examine each annotated set over all three data sets. First let us examine the Mohit's test sets over the N-gram data. Over all, all the clusters have some corpus evidence for at least one pair of comparisons. Note similar to Mohit's algorithm, we correctly place "first, . . ., eight" in the correct order even though we

only observe data for some of the comparisons. Furthermore we note that sometimes multiple orderings in $\mathbf{\Omega}$ will have equal probabilities, this is not the case here, we picked the unique ω^*. In the case of "close, near, intimate" we only observe data for one out of three possible comparisons, and correctly placed the near to be less than intimate. Now we have two possibilities: "near $<$ close $<$ intimate" and "close $<$ near $<$ intimate". Since the list of options are sorted lexicographically and "near" comes after "close" lexicographically, we picked "near $<$ close $<$ intimate". In "real, solemn, serious, grave", there is strong n-gram corpus evidence that serious is less intense than solemn, thus the order is flipped. Finally, 9 cluster have a negative τ, all of which have strong corpus evidence supporting the ordering under the measure we defined.

Now we examine Mohit's test over the PPDB data, note it performed only slightly better than the random baseline, suggesting the PPDB corpus has marginal value add. Notably, the cluster "first, . . ., eight" performed poorly because there is no data in the PPDB graph for these words. The cluster "close, near, intimate" now has a negative τ because there are four paraphrases in the PPDB corpus suggesting "close $<$ near":

1. quite close $\rightarrow$ near
2. real close $\rightarrow$ near
3. really close $\rightarrow$ near
4. so close $\rightarrow$ near
5. too close $\rightarrow$ near
6. very close $\rightarrow$ near,

and no evidence suggesting "near $<$ close." In the cluster "real, solemn, serious, grave" we output an ordering that is of the same pairwise accuracy and τ value as the N-gram case, but making different mistakes: flipping the order of "real" and "solemn" in this case but not because of evidence because there is none, and "solemn" is after "real" lexicographically. Overall, 47 out of the 79 clusters in the Turk set did not have any data for any pairs of comparisons, this is close to 60% of the data set. In all the clusters with a negative τ, all but one case was due to complete lack of data. A similar story is true in BCS data set, where our method performs better than the "random" baseline because every time there is evidence for the words, the $<$ sign is flipped to the correct ordering. Finally, we simply note that Mohit's data set did comparably well on the PPDB + N-gram set, this is not surprising since the many of the Mohit's words do not make an appearance in the PPDB data set.

Now we consider the Turk set over all data sets. On the N-gram data, the Turk set performed no better than random. Over 70% of the clusters in the Turk set do not have any observations in the N-gram set. Thirty five clusters in this set have negative τ's, and 29 of which are due to lack of data. In other cases, a negative τ is either due to corpus evidence contradicting the gold set (two cases) or due to ties in the gold set, which our model does not account for. Next we consider the Turk set over the PPDB data, note that 68% pairwise accuracy is a respectable showing if we recall that Mohit achieved 69.2% accuracy with the MILP formulation on his dataset. Over all 38 of the clusters have data for every possible comparison between words ($O(n^2)$ comparisons), however 23 of these clusters have only two words. Five clusters have no observation for any of the links, curiously three out of these five clusters also only contains two words. For the curious the clusters are:

1. uncomfortable, embarrassed

2. shitty, awful

3. sturdy, intact

4. vast, plenty, abundant

5. tough, formidable, daunting.

Suffice it to say the last cluster is a fair description of writing this thesis, we hope the second cluster does not describe the experience of *reading* this thesis. Finally 14 clusters have negative τ's, two of these clusters have no data (the second and third cluster from the list above, for those who are wondering). The rest have corpus evidence that contradicts annotators. For example, the annotators ranked "hardworking $<$ tough $<$ tenacious", while the algorithm ranked "tough $<$ tenacious $<$ hardworking" because in the PPDB corpus we observe:

1. very tough $\rightarrow$ tenacious

2. very tough $\rightarrow$ hardworking

3. pretty tough $\rightarrow$ hardworking

4. really tough $\rightarrow$ hardworking

5. so tough $\rightarrow$ hardworking.

We leave it to the reader's imagination for the interpretation of $<$ in this setting, and which quality should rank higher under $<$. Finally we examine the Turk set over the PPDB + N-gram data set, where our estimation performed as well as we did on Mohit's set. This is encouraging because this suggests that although the two data sets performed drastically differently over the PPDB and N-gram data sets alone, they performed comparably well on the combined data sets. Which

confirms our suspicion that the lack of data is the cause of the discrepancy, not how the sets are curated.

Finally we examine the base comparative superlative set. When test on the N-gram data set, 189 out of 285 clusters have no data according to the N-gram corpus, 64 clusters have data for every pair of comparison. One hundred and eighty two clusters have negative τ's, 24 of which have data for one comparison out of the $O(n^2)$ possible comparison between words, in all cases corpus evidence placed the words in the correct order, but lexicographical ordering ensured that the overall order of words still has a negative τ. For example, in the cluster "short, shorter, shortest" we observe $\boldsymbol{Pr}[short < shorter] = 0.99$ but have no observations for the other pairs, so the overall ranking is $shortest < short < shorter$, again due to lexicographical ordering.

Now we examine the BCS set on the PPDB data set. Pairwise accuracy is appreciably higher on this data set because only 101 clusters have no data, while 159 have data between all $O(n^2)$ possible comparisons. Ninety five clusters have negative τ's, however six of these clusters have data supporting the ranking. The ranking output by our methods are:

1. more $<$ many
2. more $<$ some
3. more $<$ much
4. better $<$ well
5. latest $<$ later
6. poorest $<$ poorer.

The fact that "more" makes such a strong showing is curious, in the first cases, there exists just one edge from "more" to "many": "any more $\rightarrow$ many." In the second case there is just one edge again: "a few more $\rightarrow$ some". In both cases there are no edges going the other way. In the third cases, there are six edges suggesting more is less intense than much:

1. a lot more $\rightarrow$ much
2. considerably more $\rightarrow$ much
3. little more $\rightarrow$ much
4. lot more $\rightarrow$ much
5. significantly more $\rightarrow$ much
6. substantially more $\rightarrow$ much,

and just one edge suggesting more is more intense than much: very much $\rightarrow$ more. A sharp eyed reader might immediately ask how connected are the two vertices relative to each other, to satiate your curiosity we report the values here: more has 147 total neighbors, 42 in-neighbors and 115 out-neighbors. Much on the other hand only has 78 neighbors, 23 in-neighbors and 66 out-neighbors. In other words, one vertex may appear to dominate another if we consider the number of edges alone, but may not dominate if we take their overall connectivity in the graph into account. We will use this particular observation to improve our results later.

Finally we close the chapter by examining the BCS labeled set on the PPDB + N-gram data. Incorporating N-gram data has the immediate consequence that one more cluster now has data for all possible comparisons among adjectives, progress

comes in the most incremental steps. Ninety nine clusters still have no data whatsoever, and 93 clusters still have negative τ's, 81 of which are because there is no data for any pairs. Only 5 of these negative clusters have data between all pairwise comparisons, they are have already been listed above. Notably, "more" is now correctly classified as less intense than "much" due to overwhelming N-gram evidence: 67386 edges point from "much" to "more", while only 2834 edges point the other way.

In conclusion, we observe three kinds of mistakes:

1. mistakes from no data between pairs
2. mistakes from data between pairs that contradicts the gold set
3. mistakes from predicting some ordering, when in fact the words are tied.

We write off mistake number two as bad luck from a poor sample. We also write off mistake three since we will not be developing models that predict synonyms, which is a different task all together. In the next few chapters, we will focus on addressing mistake one: missing data. A very important observation is that in order to increase the overall accuracy, it suffices to have a model that gives us a slight edge over random baseline most of the time in the case for when there is no data. That is we require a model so that:

- if $s > t$ then we need $\boldsymbol{Pr}[s > t] \geq \frac{1}{2} + \epsilon$
- if $s < t$, then then we have $\boldsymbol{Pr}[s > t] \leq \frac{1}{2} - \epsilon$,

with high probability.

Remark 2.2. What exactly does high probability entail in our setting? There appears no right answer here, $\frac{1}{n^2}$ in the number of pairwise comparisons may be ideal,

	N-gram			PPDB			PPDB + N-gram		
Test set	**Pairwise**	**Avg. τ**	**Avg. $\|\tau\|$**	**Pairwise**	**Avg. τ**	**Avg. $\|\tau\|$**	**Pairwise**	**Avg. τ**	**Avg. $\|\tau\|$**
Mohit	**72.0%**	**0.56**	**0.65**	46.2%	0.02	0.46	71.3%	0.53	0.66
Turk	47.0%	0.04	0.62	68.5%	0.49	0.70	**71.0%**	**0.55**	**0.72**
BCS	38.0%	-0.24	0.92	65.5%	0.30	0.94	**66.0%**	**0.33**	**0.95**

Table 2: Results across all datasets. Observe how N-gram graph only performed slightly better than the base line on base-comparative-superlative dataset. A similar story holds for the Turk set. However on Mohit's set we already manage to achieve a higher or comparable accuracy across all measures on the N-gram set than what Mohit did in his TACL paper.

but perhaps is too much to ask. Next we could ask for $\frac{1}{n}$, that is in a data set with 300 unknown pairwise comparisons, we need to achieve 96% pairwise accuracy. Again speaking from intuition this is a very tall order. Thus we will aim for $\frac{1}{log(n)}$, so for 300 unknown pairs we need to achieve 84% pairwise accuracy. This will be the hurdle we aim to achieve for the rest of the thesis. So in a sense, we are actually aiming for "reasonable probability" (my definition, not found in literature), not "high probability."

3 Regression

3.1 Introduction

In the previous two chapters we attempted to construct generic measures that may work for every graph, but in the end certainly did not perform on this graph. In this chapter we will construct a measure specifically for this dataset. Towards this end, we will use elastic net regression to learn what it means for s to be less than t. We construct a variety of feature representations using adverbs and/or phrases that

co-occur with the adjectives for this task, and divide our annotated data into train, validation, and test sets to assess their efficacy.

3.2 Data Set

Recall we have three sources of comparisons, N-gram data alone, PPDB data alone, and the combination of N-gram and PPDB data. For each data set, we extracted the pairs of adjectives where no data exists in corpus. The number of pairs for each annotated set for each data set is displayed in table 1. We will learn a different model for each data set (N-gram, PPDB, PPDB + N-gram) and assess their efficacy using their respective validation and test sets.

3.3 Literature Review

Note in our graph we have 610 adverbs and phrases, and since we are interested in representing an adjective using its co-occurrence with adverbs/phrases, the feature representation could be large than the number of examples. Furthermore, this representation will be very sparse because most adjectives do not co-occur with most adverbs/phrases at all. Regularization will be necessary to prevent over-fitting. We consider two regularized models: l_1/l_2-penalized regression, and elastic net regression. In this section we will give a brief review of the two models.

Elastic net regression is a natural fit for our setting because it gives us the ability to select features and control sparsity. In this section we will give a brief overview of elastic net regression. Suppose our data set is $(\boldsymbol{X}, \boldsymbol{y})$ so that $\boldsymbol{X}$ is the $n \times p$ design matrix, where each input x is represented by the appropriate feature vector $\boldsymbol{x} \in \mathbb{R}^p$, and let $\boldsymbol{y}$ be the n-dimensional response vector $\boldsymbol{y} = (y_1, \ldots, y_n)$. We assume $\boldsymbol{y}$ is generated by this process:

$$\boldsymbol{y} = \boldsymbol{X}\beta + z,$$

where z is a zero mean Gaussian noise factor. Then elastic net regression will recover the estimated $\widehat{\beta}$ where:

$$\widehat{\beta} = \arg\min_{\beta} \|\boldsymbol{y} - \boldsymbol{X}\beta\| + \lambda_2 \|\beta\|_2^2 + \lambda_1 \|\beta\|_1 .$$

Roughly, the l_1 penalty encourages a sparse solution where only a few variables in $\boldsymbol{x}$ participate in predicting $\boldsymbol{y}$, while the l_2 penalty encourages "grouping" so that more than a few variables in $\boldsymbol{x}$ participates.

Now we review logistic and regularized logistic regression for binary outcomes. Given a $n \times p$ design matrix $\boldsymbol{X}$, logistic regression models the vector of probability $\boldsymbol{p}$ by:

$$\log \frac{\boldsymbol{p}}{1-\boldsymbol{p}} = \boldsymbol{X}^T\beta,$$

and we see that $\boldsymbol{p}$ is:

$$\boldsymbol{p} = \frac{exp(\boldsymbol{X}^T\boldsymbol{\beta})}{1 + exp(\boldsymbol{X}^T\boldsymbol{\beta})}.$$

Again given the binary outcome vector $\boldsymbol{y} \in \{0,1\}^n$, the loss function $\mathcal{L}$ is:

$$\mathcal{L}(\boldsymbol{\beta}) = \log \boldsymbol{p} + (1-\boldsymbol{y})^T \log(1-\boldsymbol{p}),$$

we can find the best $\boldsymbol{\beta}$ by:

$$\widehat{\boldsymbol{\beta}} = \arg\min_{\boldsymbol{\beta}} \mathcal{L}$$

In our setting, we experiment with two penalties on $\boldsymbol{\beta}$: l_2-penalty or ridge logistic regression, and l_1-penalty or LASSO logistic regression. In ridge logistic regression, we simply add the l_2-penalty to the objective function:

$$\mathcal{L}_{ridge}(\boldsymbol{\beta}) = \mathcal{L} - \frac{1}{2}\lambda \|\boldsymbol{\beta}\|_2^2 .$$

Similarly, for LASSO logistic regression, the objective function is:

$$\mathcal{L}_{LASSO}(\boldsymbol{\beta}) = \mathcal{L} - \frac{1}{2}\lambda \|\boldsymbol{\beta}\|_1^2 .$$

Again the ridge penalty encourages grouping of all variables, while LASSO encourages sparsity.

3.4 Problem Formulation

Now we will formulate our problem in terms of the two models we just introduced. In our setting, we will define:

$$y = \begin{cases} 1 & s < t \\ 0 & otherwise. \end{cases}$$

And for each pair of adjectives s and t. Additionally, we will need to find a corresponding feature representation so that:

$$\boldsymbol{x} = g(\phi(s), \phi(t)),$$

for some function ϕ and $g : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}^p$, where $m \leq p$. In a departure from notation of previous chapter, s now refers to the string representation of the word, while $\phi(s)$ is the corresponding vector representation. If we have this model $\widehat{\beta}$ and the appropriate g and ϕ, then we can use this definition.

Definition 3.1. Given words s and t, and their representation $\boldsymbol{x} = g(\phi(s), \phi(t))$, and let:

$$\hat{y} = \widehat{\beta}^T \boldsymbol{x},$$

where $\widehat{\beta}$ is the elastic net model, then we can define:

$$\boldsymbol{Pr}[s < x] = \begin{cases} \frac{1}{2} + \epsilon & \hat{y} < \delta \\ \frac{1}{2} - \epsilon & otherwise, \end{cases}$$

for an appropriate threshold δ. Again we discard the value of $\hat{y}$ and define the probability by fiat. In the case of penalized logistic regression, we use the actual probability output by the model.

3.5 Feature Representations

In this section we describe two broad sets of features we use, and their associated function g. In the first set of features, we will represent the adjective by the frequency of adverbs incident and/or outgoing from this adjective. In the second set of features we will represent the adjective by adjectives that are its neighbors. In an attempt to avoid confusion, we will denote the first set of features $\phi(s)$, while the second set will be $\nu(s)$.

First we list the ϕ's.

1. In-neighbor only. So that for each adverb v:

$$\phi(s)_v^{in} = \begin{cases} n & \text{there are n edges pointing to } s \text{ from all neighbors with the adverb v} \\ 0 & otherwise. \end{cases}$$

2. Out-neighbor only. So that for each adverb v:

$$\phi(s)_v^{out} = \begin{cases} n & \text{there are n edges pointing from } s \text{ to all neighbors with the adverb v} \\ 0 & otherwise. \end{cases}$$

3. Concatenation of in and out neighbor. So that $\phi(s) = (\phi^{in}, \phi^{out})$. In this case we also considered $\phi(s) = (\phi^{in}, -\phi^{out})$, where $-\phi^{out}$ is scalar multiplication of -1 with all entries of ϕ^{out}.

4. Element wise addition of in and out neighbor. So that $\phi(s) = \phi^{in} + \phi^{out}$.

5. Element wise subtraction of in and out neighbor. So that $\phi(s) = \phi^{in} - \phi^{out}$.

Furthermore, for each ϕ^{in} and ϕ^{out}, we can vary the number of adverbs in the vector. We sort the adverbs by frequency of appearance and pick the top 10, 20, 30, 40, 50, 100, 200, 300, 400, 500, and all 605 adverbs/phrases.

Now we consider the ν's. In this case we find all neighbors of s and represent each neighbor by the frequency of adverbs between the neighbor and our vertex. We list the ν's.

1. In-neighbor only. So that for each neighbor t:

$$\nu(s)_t^{in} = \begin{cases} n & \text{there are n edges pointing to } s \text{ from } t \\ 0 & otherwise. \end{cases}$$

2. Out-neighbor only. So that for each neighbor t:

$$\nu(s)_t^{out} = \begin{cases} n & \text{there are n edges pointing from } s \text{ to } t \\ 0 & otherwise. \end{cases}$$

3. Concatenation of in and out neighbor. So that $\nu(s) = (\nu^{in}, \nu^{out})$. Again we also experiment with $\nu(s) = (\nu^{in}, -\nu^{out})$.

4. Element wise addition of in and out neighbor. So that $\phi(s) = \phi^{in} + \phi^{out}$.

5. Element wise subtraction of in and out neighbor. So that $\phi(s) = \phi^{in} - \phi^{out}$.

6. Bernoulli representation of in and out neighbor. So that for each neighbor t we have:

$$\nu(s)_t^{\mathcal{B}} = \begin{cases} \frac{|\{(t,s) \in \boldsymbol{E}\}|}{|\{(t,s) \in \boldsymbol{E}\}| + |\{(s,t) \in \boldsymbol{E}\}|} & \text{there is at least one edge from } t \text{ to } s \\ \frac{1}{10} & otherwise. \end{cases}$$

In all cases, we sort the neighbors so that the most connected neighbor to s is at the top, and so on. Finally we take the top n neighbors of s as its feature representation, where n is also an experimental parameter. If an adjective has less than n neighbors, then we pad the vector with 0. Note how we smooth out the

definition of ν above so that the padding can be distinguished from a case where there are vertices from s to t, but not vice versa.

Finally, we explore a variety of g's:

1. element wise addition. $\boldsymbol{x} = \phi(s) + \phi(t)$,
2. element wise subtraction. $\boldsymbol{x} = \phi(s) - \phi(t)$,
3. concatenation. $\boldsymbol{x} = (\phi(s), \phi(t))$,
4. dot product. $\boldsymbol{x} = \phi(s) \cdot \phi(t)$,

and similarly for ν. After g has been applied, we also experimented with using the raw counts, versus normalizing the entries of ϕ so they are between 0 and 1. Normalization appears to make sense when the N-gram data is combined with the PPDB data, since the N-gram data is often times three orders of magnitude larger than the PPDB data. We normalized the raw $n \times p$ design matrix with n samples and p features in one of two ways.

1. Normalize by mean and variance. Suppose each feature is a unique multinomial random variable $\boldsymbol{x} \in \{1, \ldots, N\}$ for some suitable N, then we can normalize this variable by:
$$\tilde{x} = \frac{x - \boldsymbol{E}[\boldsymbol{x}]}{var(\boldsymbol{x})}.$$
2. Normalize by maximum value. So we have $\tilde{x} = \frac{x}{max(x_1, \ldots, x_n)}$

3.6 Results

In addition to varying the feature representations, we also varied the emphasis over each of the two penalty terms in the objective function. For the sake of brevity,

	Mohit	Turk	BCS
N-gram	550	586	556
PPDB	750	182	294
PPDB + N-gram	408	170	290

Table 3: The base-comparative-superlative pairs form the training set for each data set, the Turk pairs form the validations set, while Mohit's pairs will be the test set. Note in almost all cases but one, the test set is actually larger than than the training set.

	Elastic Net Regression		l_1-Logistic Regression	
Gold Set	**Pairwise**	**Avg.** τ	**Pairwise**	**Avg.** τ
BCS	77.0%	0.54	86.0%	0.72
Turk	61.0%	0.22	61.0%	0.22
Mohit	71.0%	0.44	72.0%	0.44

Table 4: . Results for the two best models. Note the performance on the validation and test sets are comparable between the two models. It is also curious to note that the models performed better on the test set than the validation set. In fact, no model performed better than 70% on the validation set.

we report results from the two best performing models in each method only: (1) l_1-penalized logistic regression model with Bernoulli representation of the top 10 most connected neighbors and (2) elastic-net regression where the feature vector is the Bernoulli representation of the top 50 most connected neighbors and. Note this model is learned using the data set with PPDB and N-gram comparisons and base-comparative-superlative gold sets, validated using the Turk set, and tested using Mohit's set.

All models were implemented using Python's Scikit-learn library. The best performing l_1-penalized logistic regression model has a constant C of 0.4. While

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
BCS	100.0%	1.00	97.0%	0.93	97.0%	0.94
Turk	78.0%	0.57	69.5%	0.38	70.0%	0.40
Mohit	84.0%	0.67	48.2%	-0.04	74.3%	0.49

Table 5: Results across all datasets for pairs of gold standards for which direct comparison exists. Note how performance is higher when using N-grams alone across all gold standards. Note how PPDB data alone fails to do better than random on Mohit's clusters, even though direct direct comparisons exists for the pairs in these clusters. Suggesting the data that exists for Mohit's pairs are too noisy to give useful information.

the best performing elastic net model has an α of 0.9, and l_1 of 0.1.

One natural question we can ask is how well the model performs on the pairs of adjective with no direct comparisons, versus how well the baseline performs on the set of pairs with direct comparisons. Table 3 displays the baseline one pairs with direct comparisons. Overall, we see that the model does 9% worse on the Turk set relative to the baseline using PPDB and N-gram data, and does only 2% worse relative to the baseline on Mohit's clusters.

4 Regression

4.1 Introduction

In this section we present a remedial solution to combine our regression models with the pointwise-estimation baseline model, and a more principled solution. Since we are combining the models, our gold standards will now be the original annotated sets composed of with size between two to ten.

4.2 Remedial Solution

First we give brief reminder for our baseline model. We defined two possible events: $\boldsymbol{\Omega} = \{s < t, s > t\}$, and after observing a sequence of comparisons between s and t: $\boldsymbol{S} = \{s < t, s < t, \ldots, s > t \ldots\}$, we can ask what is the probability that the next element we will observe is $s < t$. This is a Bernoulli distribution with parameter p and it is well known that the most likely p is simply:

$$\boldsymbol{Pr}[s < t] = \frac{|\{s < t \in \boldsymbol{S}\}|}{|\boldsymbol{S}|}.$$

In the baseline, if $\boldsymbol{S}$ is empty then we defaulted to $\boldsymbol{Pr}[s < t] = \frac{1}{2}$.

Now we present the remedial solution. Recall in the previous chapter we defined this probability value for the $\hat{y}$ output by elastic net regression:

$$\boldsymbol{Pr}[s < x] = \begin{cases} \frac{1}{2} + \epsilon & \hat{y} < \delta \\ \frac{1}{2} - \epsilon & otherwise, \end{cases}$$

while we used the actual probabilty value p output by the logistic regression model. In the remedial solution, we use the elastic definition defined above, and in the case of logistic regression, we actually discard the value of p and define:

$$\boldsymbol{Pr}[s < x] = \begin{cases} \frac{1}{2} + \epsilon & p > \frac{1}{2} \\ \frac{1}{2} - \epsilon & otherwise. \end{cases}$$

This captures our intuition that the prediction output by the model is less accruate than that of the actual data. Additionally, we also constructed a version of the Turk and Mohit's clusters where ties are removed. We reasoned that since our models are designed to predict ordering, while ties can be interepreted as synonyms, clusters generated without ties may give a more "fair" representation of how well the models perform. Results are displayed below.

	Elastic Net Regression		l_1-Logistic Regression	
Gold Set	**Pairwise**	**Avg.** τ	**Pairwise**	**Avg.** τ
BCS	90.0%	0.81	93.0%	0.85
Turk	75.0%	0.62	74.0%	0.61
Turk no-tie	81.0%	0.63	81.0%	0.62
Mohit	74.0%	0.61	74.0%	0.61
Mohit no-tie	76.0%	0.52	76.0%	0.53

Table 6: . Results for the two best models combined with pointwise estimation baseline in the remedial fashion. Note how two models performs comparable across all gold sets. In addition, not the gold clusters with no ties enjoyed a higher pairwise accuracy but suffer a lower τ value.

4.3 Solution with Beta Prior

In this section we provide a more formal variant of the remedial solution. The heart of the of the problem is that we have some prior belief about the likelihood that one adjective is weaker than another, and an updated belief after observing some data, be it direct comparison or estimation from a model. Since we are modeling each edge a Bernoulli variable with parameter θ ranging over $[0, 1]$, the prior is then a distribution over the Bernoulli θ, this is the Beta distribution. In this next few paragraphs, we give a brief overview of Beta-Binomial model, in particular how it applies to our problem.

It is well known that the prior the binomial and Bernoulli likelihood function is the Beta distribution with paramters $\beta_1, \beta_2 \in \{1, \ldots\}$, where we have:

$$\begin{aligned} \boldsymbol{Pr}[\theta|\beta_1, \beta_2] &= \frac{\theta^{\beta_1 - 1}(1-\theta)^{\beta_2 - 1}}{\int_0^1 \mu^{\beta_1 - 1}(1-\mu)^{\beta_2 - 1} d\mu} \\ &= \frac{\Gamma(\beta_1 + \beta_2)}{\Gamma(\beta_1)\Gamma(\beta_2)} \theta^{\beta_1 - 1}(1-\theta)^{\beta_2 - 1}. \end{aligned}$$

The exact form of the Γ function is beyond the scope of this introduction but

the reader may select any introductory book on statistics for a refresher.

Now after observing n coin tosses with h heads and t tails for $h + t = n$, the posterior probability over θ given some prior setting of β_1 and β_2 is:

$$\begin{aligned}\boldsymbol{Pr}[\theta|h, t\beta_1, \beta_2] &= \frac{\boldsymbol{Pr}[h|n, \theta]\boldsymbol{Pr}[\theta|, n, \beta_1, \beta_2]}{\boldsymbol{Pr}[h|n, \beta_1 + \beta_2]} \\ &\propto \theta^{h+\beta_1-1}(1-\theta)^{t+\beta_2-1},\end{aligned}$$

note the posterior distribution is also a beta distribution. Now we a have the distribution $\boldsymbol{Pr}[\theta|h, t\beta_1, \beta_2]$, we can return the the pointwise estimation setting and ask what is the likelihood the next toss lands heads, this is exactly the posteriror mean:

$$\begin{aligned}\boldsymbol{E}[\theta|h, t\beta_1, \beta_2] &= \int_0^1 \theta \boldsymbol{Pr}[\theta|h, t\beta_1, \beta_2] d\theta \\ &= \frac{\beta_1 + h}{\beta_1 + \beta_2 + n}.\end{aligned}$$

Not how the last line appeals strongly to intuition and therefore can be easily used: the expected outcome of the next toss given the prior is simply the prior tosses plus the tosses observed from data.

In our setting, we fix the ratio of β_1 and β_2 so that the prior probability is exactly $1/2$, thus reflecting our ignorance. Note this is consistent with our ad-hoc setting in the remedial solution. The exactly values of β_1 and β_2 is a hyperparameter to be tuned, in practice we set $\beta_1 = \beta_2 = 1$. The coin tosses observed from data and the model are also hyperparameters. Note the more confident we are with the data, the larger the values of h and t should be with respect to β_1 and β_2. We experimented with a variety of values, and settled on the following settings for h and t:

1. If there is an observation, then we use the raw comparison counts between the adjectives as h and t

2. If there is no obervation so we are using the model, we set h to be the probability that the model predicts less than, and $t = 1 - h$.

In informal terms, we are confident in the quality of direct comparisons, if they can be observed, and not very confident in the prediction of the model. Results for Beta-Binomal model is presented below. All in all, the best model uses the beta-binomial model to combine direct observations with $l - 1$-penalized logistic regression model, the regression model uses top the coin toss probability of 10 most connect neighbors as features. This model achieved 75% pairwise accuracy on Mohit's data set and the Turk set, and a Kendall's τ value of 0.61 and 0.62. After adjusting for ties, the pairwise accuracy on Mohit's data set was 76%, which approaches the interannotator accuracy of 78%, while the pairwise accuracy on the Turks set was 82% after adjusting for ties.

	Elastic Net Regression		l_1-Logistic Regression		MILP	
Gold Set	**Pairwise**	**Avg.** τ	**Pairwise**	**Avg.** τ	**Pairwise**	**Avg.** τ
BCS	90.0%	0.81	**93.0%**	**0.85**	18.0%	0.02
Turk	75.0%	0.62	**75.0%**	**0.62**	25.0%	0.13
Turk no-tie	81.0%	0.63	**82.0%**	**0.63**	19.0%	0.12
Mohit	74.0%	0.61	**75.0%**	**0.61**	69.6%	0.57
Mohit no-tie	76.0%	0.52	**76.0%**	**0.53**	68.0%	0.46

Table 7: . The first two columns show results for the two best models combined with pointwise estimation baseline using Beta-Binomial model. The third column displays Mohit's MILP method using N-gram data only. The results shows that l_1-logistic regression outperformed elastic net regression on most data sets by a small (possibly insignificant) margin, otherwise they are equivalent. In particular, oberve how logistic regression performs just as well on Mohit's set as it does no the Turk set. Furthermore, both model outperform MILP by a non-trivial amount on all gold sets. Finally, note how well the MILP method performs on Mohit's gold cluster, versus how poorly it performs on other gold standards.