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. In fact, very often the bottleneck is in estimating the pairwise probability, not the enumeration of Π.

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 Ranking in the Presence of Missing Data

3.1 Introduction

In the previous chapter we considered pairwise comparisons between adjectives only, when there is no data supporting the pairwise comparison, we answer with "do not know." We would like to determine some way of knowing. Towards this end, in this chapter we ask if it is possible to compare the two adjectives by *not* comparing them. In other words, suppose we have a data set where we do not ob-

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

Table 3: Number of pairs where no data exist in the graph for each gold set.

serve any pairwise comparisons between adjectives in the annotated sets, how well can we perform? We present two *heuristics* towards this question: neighborhood heuristic and shortest path heuristic. But first we need to construct the appropriate test set for our task.

3.2 Test Set

Since we are only interested in attaining a pairwise accuracy for word pairs for which no data exists, we construct a new test set using annotated gold set in the following manner: for every pair of words in each cluster where no data exist, we create a new cluster with just these two words. For example, if in a cluster: "ok $<$ good $<$ great $<$ excellent" there are no data between "ok" and "great", and "good" and "excellent", then we form two clusters in the new pairwise set "ok $<$ great" and "good $<$ excellent". Furthermore, since we have three graphs: one with only edges from PPDB, one with only edges from the N-Gram set, and their combination, we construct the aforementioned pairs for each graph. The number of pairs for each set of pairs, for each graph, is listed below.

Recall in the previous chapter we aimed for $\frac{1}{log(n)}$ pairwise accuracy, below is a table that computes this accuracy for each set. In the rest of this chapter unless otherwise noted, when we say the "test set", we mean the set of adjective pairs for which no data exists.

	Mohit	Turk	BCS
N-gram	84%	84%	84%
PPDB	85%	85%	82%
PPDB + N-gram	83%	80%	82%

Table 4: The table of "reasonable probability" hurdles we must cross.

3.3 Neighborhood Heuristic Formulation

Similar to the road map laid out in the problem formulation chapter, we suppose each edge in the graph is placed independently, and similar to the previous chapter we estimate $\boldsymbol{Pr}[s < t]$ with pointwise estimation, without considering the comparisons between s and t. In the simplest case we answer with $\frac{1}{2}$ again, this is exactly the "random" baseline we witnessed. An immediate improvement is to estimate this probability using local connectivity of each adjectives.

Suppose we wish to compare vertices s and t, then for a vertex s we can form the neighbor $\mathcal{N}$ of s without t with:

$$\mathcal{N}_{s/t} = \{x : x \in \mathcal{N}_{s/t}\}.$$

Note if t is not in the neighborhood of s then $\mathcal{N}_s = \mathcal{N}_{s/t}$. Now we make the simplifying assumption that all vertices in $\mathcal{N}_{s/t}$ are the same. Again we let $<$ be a Bernoulli variable with parameter p, since all vertices are the same, so we have for $\boldsymbol{\Omega} = \{<, >\}$ and the frequencies of $<$ and $>$ are estimated from the set: $\mathcal{N}_{s/t}$:

$$\boldsymbol{Pr}_{\boldsymbol{\Omega}}[s < t] = \frac{\{s < x \in \mathcal{N}_{s/t}\}}{|\mathcal{N}_{s/t}|},$$

and $\boldsymbol{Pr}_{\boldsymbol{\Omega}}[t < s] = 1 - \boldsymbol{Pr}_{\boldsymbol{\Omega}}[s < t]$. But could also determine $\boldsymbol{Pr}[t < s]$ by examining t. That is we can form $\mathcal{N}_{t/s}$ and for $\boldsymbol{\Omega} = \{<', >'\}$ defined where the

frequencies are estimated from $\mathcal{N}_{t/s}$:

$$\boldsymbol{Pr}_{\boldsymbol{\Omega}'}[t < s] = \frac{\{t < x \in \mathcal{N}_{t/s}\}}{|\mathcal{N}_{t/s}|},$$

where $\boldsymbol{Pr}_{\boldsymbol{\Omega}'}[s < t] = 1 - \boldsymbol{Pr}_{\boldsymbol{\Omega}'}[t < s]$. These definitions are simple but they create a very pernicious problem: the probabilities do not sum to one in every case. This is not surprising since the distributions are placed over outcomes of $<$ over different sets. If our goal is to toss a coin and place edges between vertices, then what we have defined cannot be used. One way forward is to make another simplifying assumption, all vertices in neighbor of t and neighbors of s are the same, we call this vertex z. Now we have four possibilities, $s < z < t$, $s > z > t$, $s < z > t$, and $s > z < t$. In the first two cases it is clear that either $s < t$ or $s > t$, in the next two it is not clear, so we do not consider the next two cases when computing our probability; this is in the spirit of our $argmax$ function: contradictory results are not considered. There is a natural interpretation to this formulation: we are using intermediate vertices between s and t to rank them, assuming all vertices are the same. All in all, our final data set is $\mathcal{N}_{st} = \mathcal{N}_{s/t} \cup \mathcal{N}_{t/s}$, and:

$$\boldsymbol{Pr}[s < t] = \frac{|\{s < z < t \in \mathcal{N}_{st}\}|}{|\{s < z < t \in \mathcal{N}_{st}\}| + |\{s > z > t \in \mathcal{N}_{st}\}|}.$$

Remark 3.1. If $|\mathcal{N}_{st}| = 1$ so that we have a single vertex r with edges between s and r and t and r, then the formulation above is using an intermediate word to rank the two words given. Note in this case we could also have two cases where they are both stronger than r or weaker than r, there is no information gained from these cases unless we examine r itself, as a baseline we ignore these cases.

Remark 3.2. If $\mathcal{N}_{s/t} \cup \mathcal{N}_{t/s} = \mathcal{N}_{s/t} \cap \mathcal{N}_{t/s}$, then the two words are in a community for a loose definition of community, so we are using all *informative* intermediary paths in the community to rank the two adjectives.

Remark 3.3. If $\mathcal{N}_{s/t} \cap \mathcal{N}_{t/s} = \emptyset$, then we are assuming although they do not share neighbors, the vertices in their neighbors are "similar" in strength, again for a very loose definition of strength.

3.4 Shortest Path Heuristic Formulation

The previous heuristic does a breadth first search of depth one and attempt to rank s and t using all of their neighbors. Now we consider the *smallest* set of neighbors possible: this is the shortest path between s and t. Suppose we observe the shortest path from s to t: $\boldsymbol{P}_s[t] = (s, s_1, \ldots, s_n, t)$, and similarly from t to s: $\boldsymbol{P}_t[s] = (t, t_1, \ldots, t_m, s)$, then we can express $\boldsymbol{Pr}[s < t]$ by tossing coins along these two paths.

First we need to be very clear on what the set of events we are considering, and how we need to reduce this set to fit our problem. Suppose we observe $\boldsymbol{P}_s[t] = (s, s_1, \ldots, s_n, t)$, then there are $n+2$ vertices along this path, and $n+1$ edges. Thus there are 2^{n+1} possible placement of edges between these vertices, the enumeration of all these placements *could be* the set of our elementary events $\boldsymbol{\Omega}$. For example, the following three paths are drawn from $\boldsymbol{\Omega}$:

1. $s < s_1 < \ldots < s_n < t$
2. $s > s_1 > \ldots > s_n > t$
3. $s > s_1 < s_2 > s_3 \ldots > s_n < t$.

From the first statement we can conclude $s < t$, from the second we see $s > t$. In the third path, no conclusion can be drawn. In fact, in all but two paths, there are either contradictions or inconclusive evidence based on path alone. We discard these $2^{n+1} - 2$ paths and our reduced event space is just $\boldsymbol{\Omega} = \{s < s_1 < \ldots < t, s > s_1 > \ldots > t\}$.

Definition 3.1. We say s is less intense than t through $\boldsymbol{P}_s[t]$, written $s <_{\boldsymbol{P}_s} t$, if we observe the path $(s, s_1, \ldots, s_n, t)$ implying that:

$$s < s_1 < \ldots < s_n < t.$$

Assuming the edges between all vertices are placed independently , then the probability of that $s <_{\boldsymbol{P}_s} t$ is exactly the probability of this path (properly normalized of course):

$$\boldsymbol{Pr}[s <_{\boldsymbol{P}_s} t] = \frac{\boldsymbol{Pr}[s < s_1]\boldsymbol{Pr}[s_n < t] \prod_{i\in\{1,\ldots,n\}, i<j} \boldsymbol{Pr}[s_i < s_j]}{\boldsymbol{Pr}[s <_{\boldsymbol{P}_s} t] + \boldsymbol{Pr}[t <_{\boldsymbol{P}_s} s]},$$

where:

$$\boldsymbol{Pr}[t <_{\boldsymbol{P}_s} s] = \frac{\boldsymbol{Pr}[s > s_1]\boldsymbol{Pr}[s_n > t] \prod_{i\in\{1,\ldots,n\}, i<j} \boldsymbol{Pr}[s_i > s_j]}{\boldsymbol{Pr}[s <_{\boldsymbol{P}_s} t] + \boldsymbol{Pr}[t <_{\boldsymbol{P}_s} s]}.$$

Finally for each s_i and s_j, we have:

$$\boldsymbol{Pr}[s_i < s_j] = \frac{|\{s_i < s_j \in \boldsymbol{S}_{ij}\}|}{|\boldsymbol{S}_{ij}|},$$

where $\boldsymbol{S}_{ij}$ is the set of edges between s_i and s_j:

$$\boldsymbol{S}_{ij} = \{s_i < s_j, s_i < s_j, \ldots, s_i > s_j \ldots\}.$$

Note since there exists a path through all s_i's, by construction $\boldsymbol{S}_{ij} \neq \emptyset$ for every i and j.

Next we can define a similar measure for $\boldsymbol{P}_t[s]$. But since the shortest path from s to t are not guaranteed to pass through the same vertices as from t to s, $\boldsymbol{Pr}[s <_{\boldsymbol{P}_s} t] \neq \boldsymbol{Pr}[s <_{\boldsymbol{P}_t} t]$, again we need to combine the two measures. The simplest way forward might be to define s is less than t if and only if the two measures agree, and similarly for s greater than t. Indeed this is the definition we choose.

Definition 3.2. The vertex s is less intense than t under the paths enumerated by $\boldsymbol{P}_s[t]$ and $\boldsymbol{P}_t[s]$, written $s <_{\boldsymbol{P}_{st}} t$, if and only if $s <_{\boldsymbol{P}_s} t$ and $s <_{\boldsymbol{P}_t} t$. Furthemore, $t <_{\boldsymbol{P}_{st}} s$ if and only if $t <_{\boldsymbol{P}_s} s$ and $t <_{\boldsymbol{P}_t} s$. If however we observe $t <_{\boldsymbol{P}_s} s$ and $s <_{\boldsymbol{P}_t} t$ or $s <_{\boldsymbol{P}_s} t$ and $t <_{\boldsymbol{P}_t} s$, then the results are discarded. Now if we let $p = \boldsymbol{Pr}[s <_{\boldsymbol{P}_s} t]$ and $q = \boldsymbol{Pr}[s <_{\boldsymbol{P}_t} t]$, by the law of conditional probability we have:

$$\boldsymbol{Pr}[s <_{\boldsymbol{P}_{st}} t] = \frac{pq}{pq + (1-p)(1-q)},$$

and $\boldsymbol{Pr}[t <_{\boldsymbol{P}_{st}} s] = 1 - \boldsymbol{Pr}[s <_{\boldsymbol{P}_{st}} t]$.

Remark 3.4. This measure is biased against longer paths, for example if length $\boldsymbol{P}_t[s] \neq$ length of $\boldsymbol{P}_t[s]$. But by definition this cannot be an issue since in the numerator we have pq.

Remark 3.5. A demanding reader might think discarding all $2^{n+1} - 2$ events is intellectually lazy, for example if we only observe one sign is flipped in a run of all $<$'s, can't we say something about the relationship between s and t? The answer is no unless we consider vertices for which the sign has been flipped, and their neighbors. These are not the kind of questions we can ask with the elementary game of coin tosses we have committed to play. Later in the thesis we will experiment with measures using more advanced tools for just this question.

Remark 3.6. In practice there is often no path between two vertices, in this case we define the probability that $s < t$ to be $\frac{1}{2}$.

Remark 3.7. Once again we must contend with implementation issues. In Python's networkx package, if the two vertices s and t are neighbors, then the shortest path is simply from s to t, and so far as I know there is no way to output the next shortest path unless we remove all edges between s and t. Since we are only interested in evaluating how well the algorithm does in the case where there is no direct path

among s and t, if there is a direct path we output $\boldsymbol{Pr}[s < t] = \frac{1}{2}$ by fiat. This way our baseline is not contaminated by information from direct comparisons.

Remark 3.8. The remark above is important because if there is a direct path between s and t, so that $\boldsymbol{Pr}[s <_{\boldsymbol{P}_s} t] = \boldsymbol{Pr}[s <_{\boldsymbol{P}_t} t] = \boldsymbol{Pr}[s < t] = p$, then we have:

$$\frac{pq}{pq + (1-p)(1-q)} = \frac{p^2}{p^2 + (1-p)^2},$$

which is a quadratic function in p centered at $(\frac{1}{2}, \frac{1}{2})$. That is to say if $p = \frac{1}{2}$, then our expression is an unbiased estimate of p, if $p > \frac{1}{2}$ our estimate is larger than p, for $p < \frac{1}{2}$ this estimate is smaller than p. Clearly for p equals to one or zero, it is unbiased. Overall, our estimate has a strong "overconfident" bias when s and t are neighbors.

Remark 3.9. Note although we are using the "less than" sign when denoting $s <_{\boldsymbol{P}_{st}} t$, it is technically not proper ordering. For example if we have $s <_{\boldsymbol{P}_{st}} t$ and $t <_{\boldsymbol{P}_{st}} r$, we do not necessarily have $s <_{\boldsymbol{P}_{st}} r$. In fact any time we estimate the ordering from neighboring vertices, transitivity is lost.

3.5 Results

All results are presented in table two through four below. First we direct our attention to the "uniform" baseline, recall for each pair of words we have $\boldsymbol{Pr}[s < t] = \frac{1}{2}$. Note the pairwise accuracies are slightly below 50% for Mohit's set due to the natural bias from reverse sorting, again this effect is most apparent on the base-comparative-superlative set.

Now we direct our attention two table 3, and immediately observe that it only improves the baseline slightly, with the notable exception of the Turk set on the N-gram data, where we achieved 66% pairwise accuracy. Finally on the short-path

	N-gram no data		PPDB no data		PPDB + N-gram no data	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	47.0%	-0.05	48.0%	-0.03	48.0%	-0.04
Turk	40.0%	-0.19	37.0%	-0.25	41.0%	-0.18
BCS	14.0%	-0.73	15.0%	-0.70	15.0%	-0.70

Table 5: Results across all datasets for uniform baseline. Note since $|\tau|$ will be 1.0 for all test sets, it is not reported.

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	61.0%	0.21	51.0%	0.02	62.0%	0.24
Turk	66.0%	0.31	56.0%	0.12	60.0%	0.20
BCS	55.0%	0.11	50.0%	-0.01	59.0%	0.17

Table 6: Results across all datasets for neighborhood heuristic.

heuristic, we see marginal increase in performance, again with the exception of Mohit's test set on the PPDB and Ngram data set.

All in all, none of the measures passed the baseline we laid out, therefore they are all "reasonably" bad and thus do no warrant further inspection.

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	57.0%	0.14	50.0%	-0.01	68.0%	0.35
Turk	44.0%	-0.13	56.0%	0.13	51.0%	0.01
BCS	27.0%	-0.47	56.0%	0.13	65.0%	0.30

Table 7: Results across all datasets for shortest path heuristic.

4 Ranking Using Network Centrality

4.1 Introduction

Since all heuristics measures constructed in the previous chapter failed, we will consider two measures in this chapter using more principled approaches. These measure differs notably from the previous chapter in that (1) the previous chapters considers local information, here we consider the whole graph; and (2) there is large precedent in literature for these measures.

4.2 Motivation

We hypothesize that one reason local measures fails is because it is not simply the number of edges that determines the strength of s and t. Both the relative strength of of vertices incident on s versus those of t, and the relative strength of vertices pointed to by s and t are important. For example, suppose both "good" and "great" have equal number of edges incident upon it, but great is a paraphrase of "much better", which we know to be stronger than "good", then we should rank "great" as more intense than "good". What we are interested in measuring is the "centrality" of an adjective. In fact there is a family of measures constructed for just this task, and all involves defining some random walk over the graph and computing the stationary distribution over all vertices under this walk. In this chapter we consider two in particular: PageRank and Personalized PageRank. PageRank outputs a total ordering over all vertices, ranked by their centrality measure. Personalized PageRank, on the other hand, does not assume a unique ordering of all words in the graph, instead it computes how important some vertex t is given the perspective of a particular vertex s. And note the ranking of t may be different for each s. Now we will introduce some notation and review these measures.

4.3 Notation

Consider the directed graph $\boldsymbol{G}$ with vertices $\boldsymbol{V}$ and edges $\boldsymbol{E}$, the vertices in this graph be denoted by s, t, or r. An edge from s to t will be denoted (s, t). The adjacency matrix $\boldsymbol{A}$ of the graph is defined as:

$$\boldsymbol{A}_{st} = \begin{cases} 1 & \text{if } (s,t) \\ 0 & otherwise. \end{cases}$$

Note $\boldsymbol{A}_{st}$ is the st'th entry of matrix $\boldsymbol{A}$. Furthermore, the degree matrix $\boldsymbol{D}$ is the diagonal matrix with:

$$\boldsymbol{D}_{st} = \sum_{t} \boldsymbol{A}_{st}.$$

Finally we can construct the probability transition matrix $\boldsymbol{W}$ with:

$$\boldsymbol{W} = \boldsymbol{D}^{-1}\boldsymbol{A}.$$

Note for every st'th entry in $\boldsymbol{W}$ we have:

1. $\boldsymbol{W}_{st} \in [0, 1]$
2. $\sum_t \boldsymbol{W}_{st} = 1$.

Following the tradition of [citation], we will use π_s to denote a distribution over all vertices in $\boldsymbol{G}$. We also overload the letter s to denote both the vertex in $\boldsymbol{V}$ and a distribution over vertices where the entry s has probability one, and all other vertices have probability zero. Now we will define PageRank and Personalized PageRank. In the interest of highlighting their similarities, we will use similar language in both definitions.

4.4 PageRank

PageRank of vertex s is the probability that a random walk of some length starting from an arbitrary vertex will terminate at s. Formally, X_i be a random variable ranging over distributions over vertices in $\boldsymbol{G}$, then the ordered sequence $(X_0, X_1, \ldots, X_L)$ is a random walk of length L starting from $X_0 = s$. L follows a geometric distribution where $\boldsymbol{Pr}[L = l] = (1-\alpha)^l\alpha$. That is to say the random walker starts at some arbitrary vertex and with probability α, transition to an out-neighbor t according to W_{st}, and with probability $1 - \alpha$, "teleport" to an arbitrary vertex on the graph. The PageRank vector π is the solution to the expression:

$$\pi = \widetilde{\boldsymbol{W}}\pi,$$

where $\widetilde{\boldsymbol{W}}$ is the "Google" matrix:

$$\widetilde{\boldsymbol{W}} = \alpha\boldsymbol{W} + (1-\alpha)\frac{1}{n}\boldsymbol{J},$$

and $\boldsymbol{J} = \mathbf{1} \cdot \mathbf{1}^T$ is the all one matrix. Note the expression $\alpha\boldsymbol{W}$ above captures the random surfing behavior, while the second expression is the teleportation factor. The PageRank of a vertex s is the probability this random walk terminates at s: $\pi[s] = \boldsymbol{Pr}[X_L = s]$

4.5 Personalized PageRank

A personalized PageRank of vertex t relative to s is defined as the probability that a random walk of the appropriate length starting from s will terminate at t. Similar to PageRank, let X_i be a random variable ranging over distributions over vertices in $\boldsymbol{G}$, then the ordered sequence $(X_0, X_1, \ldots, X_L)$ is a random walk of length L starting from $X_0 = s$. L follows a geometric distribution where $\boldsymbol{Pr}[L = l] = (1-\alpha)^l\alpha$. In other words, the random walk starts at s and with probability

$1 - \alpha$ continue to a random neighbor of the current vertex; and with probability α terminates at the current vertex s. Again at each vertex s, the random neighbor t is chosen with probability $\boldsymbol{W}_{st}$. The personalized PageRank vector π_s with respect to s is the solution to the expression:

$$\pi_s = \alpha s + (1 - \alpha)\pi_s \boldsymbol{W}.$$

The PPR of vertex t with respect to s is the probability we terminate at t:

$$\pi_s[t] = \boldsymbol{Pr}[X_L = t].$$

Solving for π and π_s is well studied but beyond the scope of this thesis, the interested reader should refer to [citation] for a thorough treatment.

4.6 Pairwise Comparison Using Centrality Measures

In this brief section we define how to assign $\boldsymbol{Pr}[s < t]$ using the two centrality measures defined above.

Definition 4.1. Given two adjectives s and t, s is less intense than t under PageRank over the graph if $\pi[s] < \pi[t]$. Furthermore by fiat, we have for some suitable ϵ:

$$\boldsymbol{Pr}[s < t] = \begin{cases} \frac{1}{2} - \epsilon & \pi[s] < \pi[t] \\ \frac{1}{2} + \epsilon & otherwise \end{cases}$$

Definition 4.2. Given two adjectives s and t, s is less intense than t under Personalized PageRank over the graph if $\pi_s[t] > \pi_t[s]$. And similar to above, we can define:

$$\boldsymbol{Pr}[s < t] = \begin{cases} \frac{1}{2} - \epsilon & \pi_s[t] > \pi_t[s] \\ \frac{1}{2} + \epsilon & otherwise \end{cases}$$

Note in both cases we "forget" the actual probability induced by the random walk, and define the probability by fiat. This is due to our previous claim in the baseline chapter that a slight edge suffices. The second reason is mostly practical: due to the quality of our data, in practice it is prudent to not assign too much credence to the values output by the centrality measures beyond the gross direction they indicate.

4.7 Constructing the Transition Matrix

This section outlines two ways of constructing the transition matrix $\boldsymbol{W}$ from the multi directed graph. We only consider one variation here: whether to include the number of edges between vertices. Let us denote $\widetilde{\boldsymbol{E}}$ as the set of edges in the multi-directed graph so that $(s, t, v) \in \widetilde{\boldsymbol{E}}$ signifies an edge from s to t labeled with adverb v. In the first construction, we ignore the number of edges between two vertices s and t, and define the probability of transitioning from s to t as one over the number of out-neighbors of s:

$$\boldsymbol{W}_{st} = \frac{1}{|\{\mathbb{1}_{(s,t,v)} : (s, t, v) \in \widetilde{\boldsymbol{E}}\}|}. \tag{1}$$

We will call this construction the out-neighbor construction. In the second construction it is the number of edges from s to t, over the total number of out edges of s:

$$\boldsymbol{W}_{st} = \frac{|\{(s, t, v) \in \widetilde{\boldsymbol{E}}\}|}{|\{(s, t, v) : (s, t, v) \in \widetilde{\boldsymbol{E}}\}|}. \tag{2}$$

This construction will be referred to as the out-edge construction

4.8 Results

Similar to the previous chapter, we test on the set of pairs for which no data exists. And once again we remind the reader that we wish to be in the low to mid 80 percent for pairwise accuracy across all test sets. The results are reported in the tables below. We varied the teleportation constant α from 0.1 to 0.9 in 0.1 increments. In practice we noted $\alpha = 0.2$ performed best for PageRank, and $\alpha = 0.8$ performed best for Personalized PageRank. In the interest of space only results from the best α are reported. Scanning across the tables, we see that the centrality measures do outperform the random baseline, but their performance as a whole do not meet our minimum 80% criteria. However, the two measures performed significantly differently across the data sets, and this difference merit some brief comments. The two most salient conclusions are:

1. PageRank out performed Personalized PageRank as a whole, falsifying our hypothesis that assuming there is no total ordering over adjectives would improve results.

2. out-edge construction out performed out-degree construction in some cases but not others, suggesting that the exact construction of the adjacency matrix does not matter.

Detailed comments are found underneath each table. All in all, PageRank appears to show promise on some labeled sets, while Personalized PageRank does not perform. Overall, both measures are unsatisfactory.

	N-gram no data		PPDB no data		PPDB + N-gram no data	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	47.0%	-0.05	48.0%	-0.03	48.0%	-0.04
Turk	40.0%	-0.19	37.0%	-0.25	41.0%	-0.18
BCS	14.0%	-0.73	15.0%	-0.70	15.0%	-0.70

Table 8: Results across all datasets for uniform baseline coupled with reverse lexicographic sorting, reprinted here for convenience.

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	48.0%	-0.03	**64.0%**	**0.28**	55.0%	0.10
Turk	**57.0%**	**0.13**	53.0%	0.05	52.0%	0.04
BCS	73.0%	0.45	**77.0%**	**0.54**	70.0%	0.39

Table 9: PageRank using graph constructed using out-edge expression. Note BCS performed best on PPDB, this is not surprising since this is how the PPDB dataset is generated. The Turk set performed best on the N-gram set, which is curious since the Turk set was constructed with respect to the PPDB adjective set. Finally we see Mohit's set performed best on the PPDB data set, but it appears as if there is a slight negative relationship between PageRank values and adjective strength on the N-gram set.

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	49.0%	-0.02	63.0%	0.25	**59.0%**	**0.19**
Turk	**58.0%**	**0.16**	51.0%	0.01	54.0%	0.08
BCS	**72.0%**	**0.43**	63.0%	0.25	72.0%	0.45

Table 10: PageRank with out-neighbor construction. Note in theory this construction should be more susceptible to "confusing" signals due to polysemy, however in practice it actually performs better on some annotated sets.

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	60.0%	0.20	39.0%	-0.22	63.0%	0.25
Turk	35.0%	-0.30	62.0%	0.23	58.0%	0.15
BCS	29.0%	-0.42	61.0%	0.22	57.0%	0.14

Table 11: Personalized PageRank with out-edge construction. Note it performs significantly worse than PageRank. In particular there appears to be a negative relationship between PPR and adjective strength on some data sets, and positive relationship on others. Furthermore, observe the negative correlations appear on the N-gram dataset where we do not observe paths over several vertices, this makes PPR highly unreliable. It also appears for Mohit's set on the PPDB data, which was not constructed with Mohit's gold set in mind.

	N-gram		PPDB		PPDB + N-gram	
Test set	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**	**Pairwise**	**Avg. τ**
Mohit	61.0%	0.22	39.0%	-0.21	62.0%	0.24
Turk	34.0%	-0.31	59.0%	0.19	49.0%	-0.01
BCS	32.0%	-0.37	63.0%	0.25	68.0%	0.35

Table 12: Personalized PageRank with out-neighbor construction. Note in general it is not worse than the out-edge construction. Finally, the negative correlation appear here in the same test set and gold set.

5 Regression

5.1 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.

5.2 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.

5.3 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. Thus regularization will be necessary to prevent over-fitting. 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.

5.4 Problem Formulation

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

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

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.

5.5 Feature Representations

In this section we describe the set of features and g we use. We construct several ϕ's:

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

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

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

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

3. concatenation of in and out neighbor. So that $\phi(s) = (\phi^{in}, \phi^{out})$,

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

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

Table 13: 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.

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. 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. $\boldsymbol{x} = \phi(s) \cdot \phi(t)$.

5.6 Results

In addition to varying the feature representations, we also varied the emphasis over each of the two penalty terms in the objective function. All results are displayed in the tables below. Before a more nuanced discussion, we will highlight a few points:

	N-gram			PPDB			PPDB + N-gram		
Test set	**Pairwise**	**Avg.** τ	**Avg.** $\lvert\tau\rvert$	**Pairwise**	**Avg.** τ	**Avg.** $\lvert\tau\rvert$	**Pairwise**	**Avg.** τ	**Avg.** $\lvert\tau\rvert$
Mohit	00.0%	0.00	0.00	00.0%	0.00	0.00	00.0%	0.00	0.00
Turk	00.0%	0.00	0.00	00.0%	0.00	0.00	00.0%	0.00	0.00
BCS	00.0%	0.00	0.00	00.0%	0.00	0.00	00.0%	0.00	0.00

Table 14:

- Once again none of the methods passed the $\log(n)$ threshold we wished. In fact, no method performed above 71% accuracy.

-