Classification and visualisation of politeness

Carla Groenland
10208429

Harrie Oosterhuis
10196129

Joost van Doorn
10805176

Ties van Rozendaal
10077391

Abstract

In this report we propose a method for classifying the politeness of language. We use annotated data gathered from Stack-Exchange and Wikipedia which was divided into polite, neutral and impolite instances. Several approaches were investigated: using Bag of Words with plain words, with words and Part Of Speech-tags, Data Oriented Parsing based features and a combination of the three. Furthermore, we propose a method for reducing the dimensionality of the data based on the information gain of individual features. The four feature-spaces perform comparably well, with the combination of all features giving slightly higher results. The results of dimensionality reduction gives intuitive results which provide a small insight into politeness in the English language.

1 Introduction

When writing an email to your supervisor, you might be wondering 'Am I being sufficiently polite?' or 'How could I make make the email more polite?'. This report evaluates a system that was built in order to be able to answer such questions. The goal is hence two-sided:

- Building a classifier for politeness.
- Finding out what it is about a sentence that makes it polite or impolite.

We compare two types of features for classification: words (with POS-tags) and DOP-features (Section 2.4). We have used the Stanford Politeness Corpus [2] as data set, which contains sentences from Stack-Exchange and Wikipedia with annotated politeness scores. The scores were obtained by using a function of the politeness scores of the sentence as assigned by five different people. For our purposes, we have converted the numerical scores to three classes. Using this dataset and our features, we can then build a classifier that labels new sentences on their politeness. In order to find out which features are important for politeness we look at their information gain (Section 2.6). Furthermore, we look at two approaches for labeling individual words in a sentence. Firstly, using politeness scores for entire sentences we compute a mean politeness scores for a word by taking the mean score of all sentences that contain the word. Secondly, we made a topic model that has the labels for the words as a latent variable which we assign using Gibbs sampling (Section 2.7). The resulting application can be found on `joostvandoorn.com/politeness/`.

2 Method

2.1 Dataset

The dataset we use is the Stanford Politeness Corpus [2], which contains sentences with their politeness score. As mentioned in the introduction we converted the scores into three different classes,

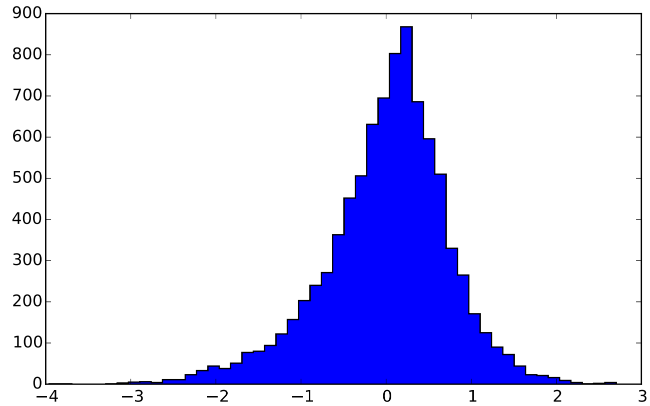


Figure 1: Distribution of scores in the Stanford Politeness Corpus

which are impolite, neutral, and polite. The score is rather arbitrary as can be inferred by the large disagreement between the annotators described in Section 3.1. Therefore there is also no direct translation between the scores and a certain class label, consequently we decided to split the data in equal portions. However due to the distribution of the scores, which can be seen in Figure 1, most sentences are neutral. We split the data into 25% impolite, 50% neutral, and 25% polite sentences, of which subsequently half of the neutral sentences were discarded to get an equal distribution of the classes. This resulted in a split points of -0.39 and 0.45 for the politeness score in the Stanford Politeness Corpus.

2.2 Basic model

Before creating our baseline we took a simple, intuitive, and naive approach. Based on the training data, all words are given the mean score of all sentences containing the word as score. The score for a test sentence is computed by summing the scores of all words that occur in the sentence, where unseen words are ignored. Words with a mean score between -0.5 and 0.5 were ignored. The scores were converted to -0.5 and 0.5 split, into impolite, neutral, and polite classes. This method achieved an accuracy of 47% on the test set.

2.3 Bag of words

As baseline we have used a bag of words (BoW) model. The occurrences of each word in a sentence are counted, and used as features for the model. The intuition of this model follows the fact that certain words are mostly used in polite sentences, and others are more common in impolite sentences. For example “Would” and “Could” are generally only used in polite sentences, and “Will” and “Can” are generally considered less polite than the former. We extended the baseline model with Part of Speech (POS) tags: instead of keeping counts for just the words, we kept counts for the words with their POS tag. The use of POS tags can disambiguate different uses of a word. Due to the fact that the BoW model does not take into account word order, most impolite sentences which use polite words, irony or sarcasm are likely to be misclassified.

2.4 DOP features

A common recommendation by English style guides is using indirectness and specific use of tenses for polite language. BoW with Words and POS-tags are inappropriate for representing such structural features, since except for specific words (for instance, *would* or *could*) no structural elements are represented. For this reason, we have also tried a representation using Data Oriented Parsing-features. Data Oriented Parsing (DOP) [4] is a probabilistic model that takes into account all the subtrees of a sentence's parse tree. We have used techniques designed for discontinuous DOP to extract subtrees for each sentence in the data. The intuition is that these subtrees can better represent structural aspects of polite language, since one might expect the directness of a sentence to be more easily identifiable by its parse tree than by its individual words. Accordingly, we have defined a new feature space where each feature represents the number of times a subtree occurs in a comment. However, since the number of subtrees in a datasets grows exponentially with its size, subtrees were selected using double-DOP. This means all subtrees that occur twice or more in the dataset are considered, and all other trees are discarded in the feature-representation. The result is a dataset where each comment is represented by a feature vector based on its subtrees which can then be used with regular classification methods.

In order to find the appropriate parse-trees for each comment we used the BLLIP-parser [3]. For each sentence the parse-tree was generated from which subtrees were extracted using an online available implementation by Andreas van Cranenburgh[1]. The average number of sentences per comment was approximately 2.13. The feature space was based on the training set, such that any subtree that appears in the test set but not in the training set is discarded. The resulting feature space consists of 261395 subtrees. Section 2.6 explains how we dealt with this large amount of features.

2.5 Classifiers

Initial experiments with numerous well known classifiers were performed to profile which classifiers appeared the most promising. This resulted in the expectation that the Support Vector Machine (SVM), Multilayer Perceptron (MLP), Naive Bayes and Linear Regression are best suited for our problem. For this study the implementation of these classifiers from scikit-learn[2] was used. In order to maintain a balance within the data, the classes were either weighted or instances were discarded. This prevents the classifiers from being biased in favor of the most represented class. In our case, a classifier could always classify an instance as neutral and achieve an accuracy around 50%, but by balancing the data such a strategy would only result in an accuracy of 33%.

2.6 Information gain as importance measure

When using words and parts of parsing trees as features for classification, the number of features is going to be significantly larger than the number of training sentences. For comparison, our training set consists of less than 9000 examples, whereas over 260.000 features are extracted from these using DOP.

In order to improve classification results and in order to better understand how politeness is caused, we ordered the features based on their *information gain*. The information gain is obtained by taking the difference in entropy of the data set before and after splitting based on whether a training example contains the feature. Entropy measures the randomness in a system, in such a way that an increase in entropy should correlate with a discriminative feature for politeness. An extreme example would be a feature that only occurs in polite sentences, such a feature would be very useful for classification and would also be interesting for understanding politeness better. The formula for entropy is

$$H(\{x_i\}_{i=1}^n) = -\sum_{i=1}^{n} p(x_i)\log(p(x_i)),$$

which would be maximal when all probability is assigned to a single state x_i (for example, when all sentences with our special feature are polite). Information gain is then defined by

$$IG(feature) = H(feature)p(feature) + H(\neg feature)(1 - p(feature)) - H(original),$$

the difference in entropy before and after splitting on the feature, where the entropy of the partitioned sets are weighted using the size of the sets. A feature that almost never appears hence gains a lower information gain, which is desirable for implementation purposes.

2.7 Topic model and Gibbs sampling

Let N sentences be given containing words $w_{n1}, \ldots, w_{nM_n}$ for $n = 1, \ldots, N$. Each word w_{nm} is generated from a distribution of words ϕ_z, with the possibilities background (neutral), polite

[1] https://github.com/andreasvc/disco-dop

[2] scikit-learn.org

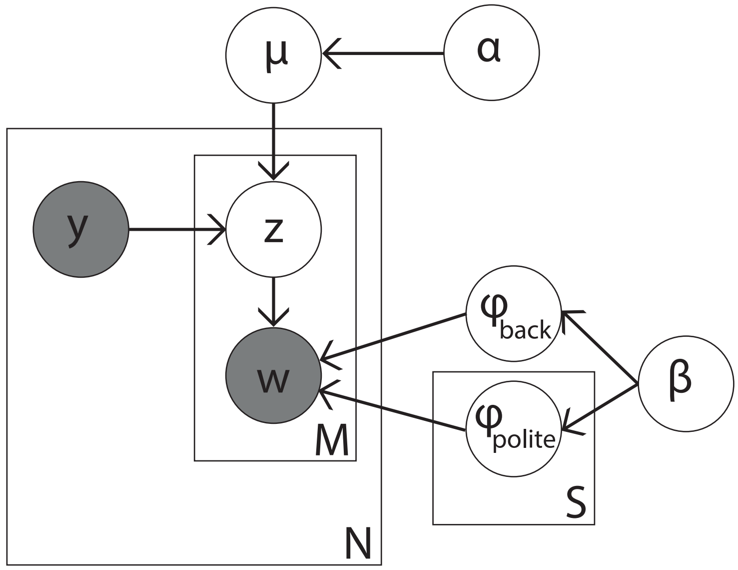


Figure 2: Graphical model for our topic model.

or impolite. These ϕ_z are Dirichlet distributions with hyperparameter β. In order to see which distribution a word is generated from, we attach a (latent) variable z_{nm} to each word. Furthermore, a sentence as a whole can be either polite or impolite, which we store in the (observed) variable y_n.

We make the assumption that polite sentences only contain words that were generated from the polite distribution or the background distribution, where the distribution over both distributions is specified by μ. Similarly, μ is used for impolite sentences to give the distribution over impolite words and backgrounds words. The distribution given by μ is a Beta distributions with hyperparameter $\alpha = (\alpha_1, \alpha_2)$. The generation of a topic z_{nm} for word w_{nm} is hence sampled from μ, where y_n picks either the polite or the impolite distribution.

In order to visualize which words are polite, which are impolite and which are background, we want to sample the topics z_i for the words w_i of a new sentence. For this purpose, the sentence first needs to be classified as either polite or impolite (which gives the value for y) and then we need to sample from $p(z_i|W, y, \Phi, z_{-i}, \mu)$, where W consists of all words in the sentence, z_{-i} are the other topics and $\Phi = \phi_1, \phi_2, \phi_3 = \phi_{background}, \phi_{polite}, \phi_{impolite}$.

The derivation of the form of the distribution is given in the appendix, with the result

$$p(z_{nm} = i|Z_{-nm}, W_{-nm}, \Phi, \mu) \propto \frac{\alpha_\delta + C(\delta) - 1}{\alpha_1 + \alpha_2 + C(0) + C(1) - 1} \times \frac{\beta + C_i(w_{nm}) - 1}{-1 + V_i\beta + \sum_{w=1}^{V_i} C_i(w)}$$

for $\delta = \delta_{i,\text{(im)polite}}$, V_i the number of words in the vocubulary of ϕ_i, $C(0), C(1)$ the number of background respectively (im)polite words and $C_i(w) = \#\{(a, b) : w_{ab} = w \text{ and } z_{ab} = i\}$. Gibbs sampling [1] is applied to generate topics for the words, where the topics (z_{nm}) corresponding to words in the dataset are initialized using the mean politeness score of the words. We ran Gibbs sampling for many iterations on the training set and save the corresponding counts. For a new sentence, we classify the sentence as polite or impolite and pick the most probable label for the words according to the model. Such a MAP estimate will be approximately correct now, since the choice of label for the individual words in the sentence will affect each other only due to change in distribution of the counts. If the counts are already high, adding one to one of the counts will make hardly any difference, such that the choice of labels for a new sentence is almost independent.

If the priors α, β are chosen high, the result will be a flat distribution. This can also be seen from the formula above: if α, β are chosen higher, the counts will be less important and the distribution will be rather flat. By varying the ratio between α_1, α_2, we can put our prior knowledge about the ratio background-(im)polite into the system. It may be best to pick β rather small, since we may expect that words have a preference for being generated from the background, the polite or the impolite distribution. We use the parameters $\alpha = (5, 2)$ and $\beta = 0.5$ in our final implementation of the topic model.

3 Results

3.1 Data evaluation

The sentences in the data are annotated by five different people on their politeness. Politeness itself is not an easy concept to capture in words and it can be expected that a politeness score is highly subjective. Furthermore, people paid for the number of sentences they annotate may also be less diligent with providing trustworthy annotations. For these reasons, the data can be expected to be noisy such that we cannot expect to reach performance above a certain threshold.

The annotators have ranked the sentence on its politeness by giving a score between 1 and 25. We have tested each individual annotator on our classification problem (with 25-50-25 data split) to see how well they would have performed. For this purpose, we have varied the possible split points (R, L) where $x < R$ is impolite, $x > L$ is polite and in between is neutral. Using the individual annotator scores, we classified the sentences and compared this to the label that was given to the sentence. We picked the split points that gave the best result for the average classification rate. Note that the score given by an annotator also influences the label given to the sentence, such that in this set-up, the human annotator has a huge advantage over our implementations. Still, the annotators only classified 57% correctly on average (using split points (12,16)) and classified the same label only 11% of the time (with split points split points (12,16) again).

Furthermore, the performance gets worse if you average over the labels. We counted the number of polite, neutral and impolite examples and the mean number of times the derived annotator labels differed for an example with a certain label. Computing the mean accuracy per label and taking the average over these, the best split was still (12,16) but the percentage decreased to 48%.

For our purposes, it would have been better if more sentences would have been annotated. The types of data annotated are also rather limited, since only Stack-Exchange and Wikipedia comments are commented, which are a very specific type of sentences. In particular, the model will probably not learn what swearing is and a large part of the data will be neutral (technical) language. Furthermore, the data seems to be very noise, based on the performance of human annotators. This problem may be too ill-defined for classification into 25 classes (as was asked of the annotators) and the data might have been better if they were only asked to classify in three or five categories. In order to improve classification results etcetera, improving the data could make the most drastic change in our opinion. Finally, the question ”how polite is this sentence” may be too ill-defined for humans, as can be seen from the small number of times five human annotators agree on a label.

3.2 Words with the highest information gain

As explained in Section 2.6 we have ranked the features based on their information gain. In Table 1 the results for the word features can be found.

Words like “Could”, “Can”, “Would” are usually indicative of polite question whereas a sentence starting with “Why” is usually not going to be very polite. This can be seen in the table, since all four words have a high information gain and the first three have a positive mean score whether the “Why” has a negative score. Other intuitive results are that “please”, “thank” and “Good” are seen as polite by our measure and that “yourself” and “homework” are indicators for impolite sentences. The latter is rather domain-specific: people on stack-exchange do not want to be doing your homework for you. Furthermore, most of the words that are seen as least indicative for politeness are also very intuitive: “formats”, “locally”, “downloaded” are some examples of word features that would probably not be indicative for politeness.

Word	Mean score	Information gain	Word	Mean score	Information gain
Why	-0.5993	6.321E-04	My	1.199E-02	2.954E-08
Thanks	0.8064	4.810E-04	curious	3.596E-02	2.954E-08
why	-0.3158	2.093E-04	answered	4.354E-02	4.094E-08
Hi	0.476	1.828E-04	assume	4.552E-02	4.466E-08
help	0.3603	1.716E-04	most	1.259E-02	4.718E-08
thanks	0.9506	1.676E-04	suggesting	-0.1112	4.866E-08
not	-0.172	1.658E-04	downloaded	6.108E-02	4.866E-08
!	0.1226	1.456E-04	developer	9.596E-02	4.866E-08
Would	0.4526	1.446E-04	ignorance	0.1725	4.866E-08
Thank	0.982	1.441E-04	drive	-6.205E-02	4.866E-08
?	-4.073E-02	1.431E-04	insert	-4.491E-02	4.866E-08
for	0.1537	1.352E-04	Anything	8.783E-02	4.866E-08
Can	0.2124	1.281E-04	cross	-9.572E-02	4.866E-08
homework	-0.6805	1.069E-04	past	0.192	5.574E-08
I	9.294E-02	9.161E-05	likely	-2.389E-02	5.806E-08
You	-0.2605	9.153E-05	formats	-1.997E-02	6.967E-08
Could	0.236	9.123E-05	divide	0.3744	6.967E-08
could	0.237	8.397E-05	dynamically	8.206E-04	6.967E-08
good	0.2565	8.212E-05	ideal	-0.1281	6.967E-08
Good	0.7076	8.162E-05	developers	0.3626	6.967E-08
please	0.2027	7.797E-05	newbie	-0.1208	6.967E-08
really	-0.2473	7.524E-05	vertical	-1.444E-02	6.967E-08
yourself	-0.5114	6.922E-05	locally	-1.612E-02	6.967E-08
mind	0.4262	6.691E-05	invalid	-0.1295	6.967E-08
did	-0.1736	6.678E-05	tube	-0.2659	6.967E-08

Table 1: The twenty-five most (left) and least (right) informative words in the training set according the information gain.

3.3 DOP features with the highest information gain

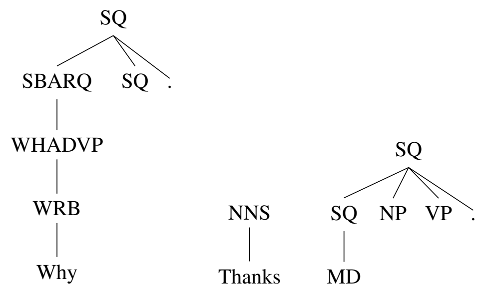
Similar to the ranking of words the DOP-features in our feature space were ranked based on their information gain. The resulting ranking showed that the top 25 most discriminative features are all variations on the three subtrees displayed in Table 2. The first feature represents a "Why" question which is usually labeled impolite, the second feature shows the usage of "Thanks" which is generally considered polite. Both these features correspond with the ranking of words, however the last feature can not be represented by BoW based on words and POS-tags. The third feature represents sentences starting with a modal verb, for instance *Would you like ...* or *Could it be that ...*. Sentences with this subtree are generally neutral or polite which is in agreement with the idea that indirectness contributes to the politeness of a sentence.

3.4 Classfication using word or DOP features

The performance of the politeness classifiers based on word counts versus classifiers based on DOP features is shown in Figure 3. For both word and DOP features, the linear SVM was the best classifier. As expected, in both cases the accuracy increased with the number of features. The optimal performance did not differ significantly between both types of features (Table 3) and occured at a similar number of features. However, classifiers based on word features only have a better accuracy when using less features. Moreover, classifiers based on DOP features have a significant drop in performance when more features are added.

3.5 Classification using both word and DOP features

The results in the previous section suggested that a linear SVM was the best classifier for this experiment. This was also suggested by pilots with different classifiers. A linear SVM was also trained with different combinations of word and DOP features, of which the results are shown in Figure 4. It can be seen that at least a hundred features are required for an accuracy of around 50%, however



Average Score:	-0.59	0.80	0.26
Polite:	5%	75%	36%
Impolite:	54%	5%	10%
Neutral:	41%	20%	54%

Table 2: The three most discriminative DOP-features in the training set according the information gain with their average score and distribution over the politeness labels. The twenty five most discriminative DOP-features are all variations on these three features.

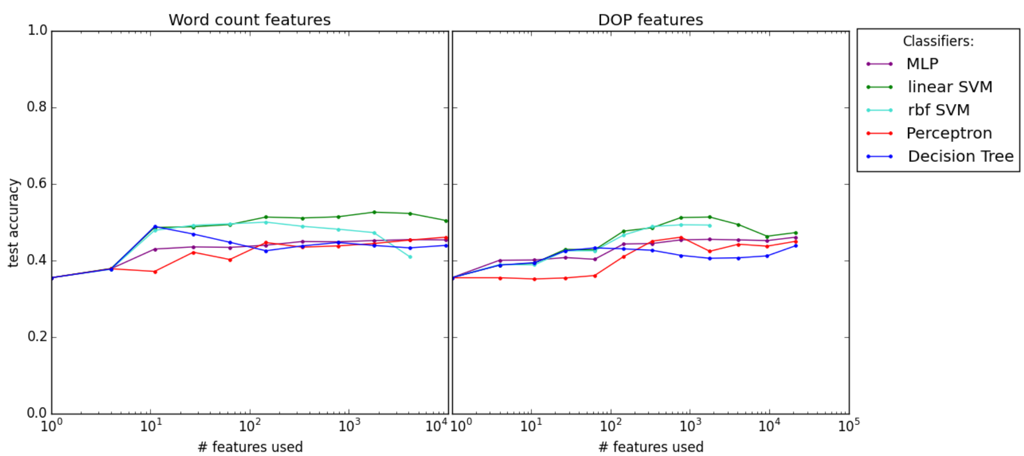


Figure 3: Performance of classifiers using either word or DOP features.

it seems words and DOP features hardly complement each other as there is no significant increase when combining them. Thus, while the highest accuracy achieved was using a combination of words and DOP features, the difference is insignificant compared to using only words or DOP features. The results of the best classifiers are repeated in Table 3.

Features used	# word features	#DOP features	Accuracy on training data	Accuracy on test data
Only word features	1779	0	0.7392	0.5263
Only DOP features	0	1779	0.6746	0.5136
Both DOP and word features	149	1779	0.8950	0.5308

Table 3: The accuracy of linear SVM using a combination of word and DOP features.

3.6 Topic model distributions

In Table 4 the results for the topic model can be found. We have given the top 10 words for each distribution, where only words that occur at least N times are considered and were the words are

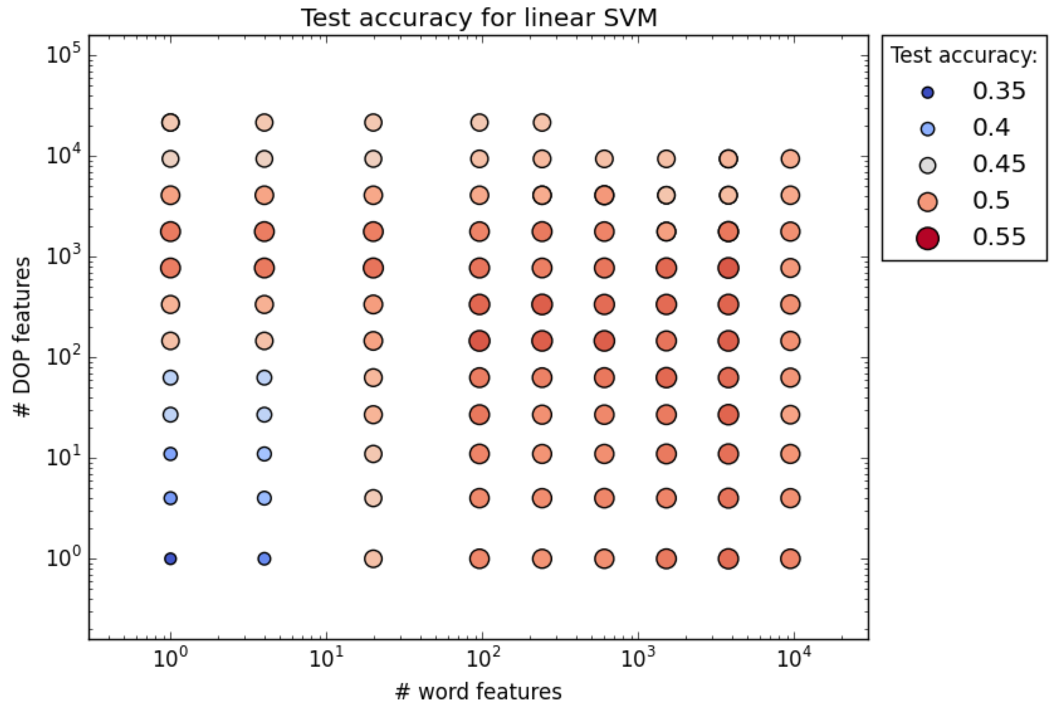


Figure 4: Accuracy on the test set of linear SVM using both word and DOP features.

scored based on the proportion of counts that lie within a certain distribution (i.e. their normalized counts over the distributions). We have considered two initializations, a random initialization and one that uses the mean score of words. In the latter case, we initialized the topic model using the the mean score of the words and the label of the sentence in the data: words with a score above 0.2 in polite sentences get a polite label, words scoring between 0.1 and 0.2 get a polite or background label assigned randomly and the remainder gets a background label. The labels for impolite sentences are assigned similarly. The top 10 before and after Gibbs sampling for ten million iterations (where one iterations is a single, random update) are given.

Only looking at words that occur often ($N = 100, 300$), the results are rather intuitive. Similarly to what was found using entropy, words like thanks, could, why and not are found on top of the (im)polite distributions. Also note the domain specific results that are occuring again, such as wikipedia in the impolite distribution and code in the polite distribution. Their does not seem to be much difference between initializing the words randomly or based on their mean scores, which is probably due to the fact that words are only allowed to get a background label or the label of the sentence, such that words with a high mean score will also have a higher expected proportion of polite distribution.

The other setting do not give these nice intuitive results, nor does Gibbs sampling (actually applying the topic model) seem to improve the results. Several flaws of our topic model may be the cause of this result. First of all, sentences are not "polite" or "impolite", but can be in an entire spectrum around those two, whereas our model enforces the sentences to be labeled in two categories. Furthermore, the assumption that a sentence may only contain polite or background words may be too restrictive. Finally, the politeness of a word can depend heavily on its context, which is not modelled in our topic model. The first two problems could easily be resolved (by adding more categories or distributions), whereas the last may be an indicator may be an indicator that a topic model as tool does not fit as a model for politeness.

Random initialization

N	Gibbs	Impolite	Polite	Background
10	before	grammar, banned, realize, faith, street, excuse, explained, hows, perfectly, claiming	manage, peer, affect, individual, construct, edge, includes, overlapping, referenced, menu	displayed, citations, night, insert, speak, requirement, prompt, respect, developer, stick
10	after	claiming, followed, banned, sorting, media, accurate, sock, attached, img, pov	subpage, displayed, session, affect, cycles, locations, sides, attribute, fa, recommend	purposes, types, methods, heard, blue, putting, released, u, enabled, column
100	before	homework, why, wikipedia, come, really, anyone, again, questions, person, exactly	thanks, id, hi, server, x, correct, each, system, add, check	given, long, their, though, help, tell, perhaps, while, cant, may
100	after	homework, seem, clear, why, understand, who, going, anyone, really, enough	thanks, start, 0, hi, created, seem, clear, back, mind, clarify	many, tag, number, source, second, non, another, post, added, provide
300	before	why, really, dont, not, youre, now, mean, no, did, answer	thanks, code, out, can, could, some, please, would, for, if	should, am, here, could, 2, thanks, im, also, see, think
300	after	why, really, than, not, mean, dont, something, no, this, where	thanks, could, 1, than, up, code, page, using, please, im	2, way, need, does, they, here, will, my, its, also

Initialization using mean scores

N	Gibbs	Impolite	Polite	Background
10	before	claiming, banned, 3rr, pov, config, followed, hell, https, excuse, virtual	subpage, displayed, cycles, session, assembly, attribute, sides, locations, loaded, tomorrow	there, de, interface, gmail, managed, played, private, term, auto, suppose
10	after	3rr, followed, https, virtual, sorting, accurate, claiming, hell, attached, attack	subpage, cycles, displayed, affect, sides, attribute, fa, recommend, vertices, intro	among, citations, disambiguation, dyk, nominated, commons, rfa, objection, edits, replied
100	before	homework, why, wikipedia, people, really, who, anyone, exactly, wrong, saying	thanks, hi, help, check, mind, may, look, could, though, good	there, back, url, articles, title, given, own, n, part, was
100	after	homework, why, seem, function, people, person, wrong, instead, he, who	thanks, mind, hi, able, may, help, running, x, id, start	edits, file, title, part, own, questions, already, provide, most, name
300	before	why, really, did, not, youre, no, dont, article, thanks, now	thanks, could, work, way, please, would, can, more, should, any	there, url, was, see, than, other, has, they, will, 2
300	after	why, really, now, dont, did, no, mean, youre, way, just	thanks, could, can, code, would, some, for, all, work, they	sure, its, from, is, you, that, as, an, in, if

Table 4: Top 10 words for the distributions of the topic model with various settings, which are explained in Section 3.6.

4 Conclusion

The results from this study show that classifying politeness is a difficult problem, as even human annotators find it hard to agree on what should be considered polite. However, we have reached accuracies of over 50% where random classification would reach 33%, therefore to some extent politeness is classifiable. Still, our results show that classifications are not very reliable.

The different feature spaces that have been investigated all score similarly well. The difference between using words, DOP features or a combination is insignificant. Thus, there seems to be no preference in what feature space to use when only considering accuracy. However, it should be noted that extracting DOP features takes a considerable amount of time compared to extracting word-based features, which could be an incentive to prefer word-based features over DOP features.

However the feature reduction based on information gain did show an advantage from using DOP over word based features. While both feature spaces gave intuitive results when looking at the most discriminative words, DOP features are able to express structural elements which the word based features can not. For instance feature reduction on DOP features shows that sentences starting with models are likely to be polite, something word based features are unable to do directly (instead the most used modals could be seen as polite). Furthermore, feature reduction is able to give an insight into polite language as its results are comprehensive for both feature spaces.

We also investigated the use of a topic model for assigning politeness labels to individual words. We assumed sentences were labeled as polite or impolite and that words in such a sentence could be either generated from the (im)polite distribution (based on the sentence label) or the background distribution. Random initialization or initialization on mean score did not give significant differences, nor did applying Gibbs sampling (using the topic model) improve the results. Only when looking at frequent words were we able to arrive intuitive results.

References

[1] Stuart Geman and Donald Geman. *Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 6(6):721-741, 1984.

[2] Cristian Danescu-Niculescu-Mizil, Moritz Sudhof, Dan Jurafsky, Jure Leskovec, Christopher Potts. *A computational approach to politeness with application to social factors*. Proceedings of ACL, 2013.

[3] Eugene Charniak, *A maximum-entropy-inspired parser*,Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference, 132–139, 2000

[4] Van Cranenburgh, Andreas and Scha, Remko and Sangati, Federico *Discontinuous Data-Oriented Parsing: A mildly context-sensitive all-fragments grammar*, Proceedings of the Second Workshop on Statistical Parsing of Morphologically Rich Languages, 34–44, 2011, Association for Computational Linguistics

Appendix: derivation of the topic model

This appendix derives the form of the conditional distributions needed for the topic model described in this report (Section 2.7). The joint is given by

$$p(Z, W, Y, \mu, \Phi|\alpha, \beta) = p(\mu|\alpha) \prod_{i=1}^{3} p(\phi_i|\beta) \prod_{n=1}^{N} \prod_{m=1}^{M} p(w_{nm}|\phi_{z_{nm}}) p(z_{nm}|\mu, y_n),$$

hence

$$p(Z, W|\alpha, \beta) = \int p(Z, W, Y, \mu, \Phi|\alpha, \beta) d\Phi d\mu =$$

$$\left(\int p(\mu|\alpha) \prod_{n=1}^{N} \prod_{m=1}^{M} p(z_{nm}|\mu, y_n) d\mu \right) \left(\int \prod_{i=1}^{3} p(\phi_i|\beta) \prod_{n=1}^{N} \prod_{m=1}^{M} p(w_{nm}|\phi_{z_{nm}}) d\Phi \right).$$

Both parts will be worked out separately.

$$\int p(\mu|\alpha) \prod_{n=1}^{N} \prod_{m=1}^{M} p(z_{nm}|\mu, y_n) d\mu =$$

$$\int \frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \mu^{\alpha_1-1}(1-\mu)^{\alpha_2-1} \prod_{i=1}^{2} \prod_{\{n:y_n=i\}} \prod_{m=1}^{M_n} p(z_{nm}|\mu, y_n) d\mu =$$

$$\frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \int \mu^{\alpha_1-1}(1-\mu)^{\alpha_2-1} \prod_{i=1}^{2} \prod_{\{n:y_n=i\}} \mu^{C(0,n)}(1-\mu)^{C(1,n)} d\mu$$

where $C(0,n), C(1,n)$ denotes the number of background or (im)polite words in sentence n respectively. This expression can be rewritten as follows:

$$\frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \int \mu^{\alpha_1-1}(1-\mu)^{\alpha_2-1} \mu^{\sum_{i=1}^{2}\sum_{\{n:y_n=i\}} C(0,n)}(1-\mu)^{\sum_{i=1}^{2}\sum_{\{n:y_n=i\}} C(1,n)} d\mu =$$

$$\frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \int \mu^{\alpha_1-1+C(0)}(1-\mu)^{\alpha_2-1+C(1))} d\mu =$$

$$\frac{\Gamma(\alpha_1+\alpha_2)}{\Gamma(\alpha_1)\Gamma(\alpha_2)} \frac{\Gamma(\alpha_1+C(0))\Gamma(\alpha_2+C(1))}{\Gamma(\alpha_1+\alpha_2+C(0)+C(1))} \propto \frac{\Gamma(\alpha_1+C(0))\Gamma(\alpha_2+C(1))}{\Gamma(\alpha_1+\alpha_2+C(0)+C(1))}$$

for $C(0)$ the total number of background tags, and $C(1)$ the total number of (im)polite tags. Before rewriting the second integral, note that we assume a fixed vocabulary of some size V_i of possible words that the ϕ_i can generate.

$$\int \prod_{i=1}^{3} p(\phi_i|\beta) \prod_{n=1}^{N} \prod_{m=1}^{M} p(w_{nm}|\phi_{z_{nm}}) d\Phi \propto$$

$$\int \prod_{i=1}^{3} \prod_{w=1}^{V_i} \phi_i(w)^{\beta-1} \prod_{i=1}^{3} \prod_{w=1}^{V} \phi_i(w)^{C_i(w)} d\Phi =$$

$$\prod_{i=1}^{3} \int \prod_{w=1}^{V_i} \phi_i(w)^{\beta-1+C_i(w)} d\phi_i = \prod_{i=1}^{3} \frac{\prod_{w=1}^{V_i}\Gamma(\beta+C_i(w))}{\Gamma(\sum_{w=1}^{V_i}\beta+C_i(w))}$$

for $C_i(w)$ the number of (n,m) such that $w_{nm}=w$ and $z_{nm}=i$.

Combining the equations above gives

$$p(Z,W,Y,\mu,\Phi|\alpha,\beta) = \prod_{i=1}^{3} \frac{\prod_{w=1}^{V_i}\Gamma(\beta+C_i(w))}{\Gamma(\sum_{w=1}^{V_i}\beta+C_i(w))} \times \frac{\Gamma(\alpha_1+C(0))\Gamma(\alpha_2+C(1))}{\Gamma(\alpha_1+\alpha_2+C(0)+C(1))}$$

such that

$$p(z_{nm}=i|Z_{-nm},W_{-nm},\Phi,\mu) = \frac{p(z_{nm}=i,Z_{-nm},W,Y,\mu,\Phi|\alpha,\beta)}{p(Z_{-nm},W_{-nm},Y,\mu,\Phi|\alpha,\beta)} \propto$$

$$\frac{\Gamma(\alpha_1+C(0))\Gamma(\alpha_2+C(1))\Gamma(\alpha_1+\alpha_2+C(0)+C(1)-1)}{\Gamma(\alpha_1+\alpha_2+C(0)+C(1))\Gamma(\alpha_1+C(0)-\delta_{i,\text{back}})\Gamma(\alpha_2+C(1)-\delta_{i,\text{(im)polite}})} \times$$

$$\frac{\Gamma(\beta+C_i(w_{nm}))][\Gamma(\sum_{w=1}^{V_i}\beta+C_i(w)-\delta_{w,w_{nm}}]}{[\Gamma(\sum_{w=1}^{V_i}\beta+C_i(w))][\Gamma(\beta+C_i(w_{nm})-1]}.$$

The equality $x-1=\frac{\Gamma(x)}{\Gamma(x-1)}$ and some simplifications give

$$\frac{\alpha_\delta+C(\delta)-1}{\alpha_1+\alpha_2+C(0)+C(1)-1} \times \frac{\beta+C_i(w_{nm})-1}{-1+V_i\beta+\sum_{w=1}^{V_i}C_i(w)}$$

for $\delta=\delta_{i,\text{(im)polite}}$, V_i the number of words in the vocubulary of ϕ_i, $C(0),C(1)$ the number of background respectively (im)polite words and $C_i(w)=\#\{(a,b): w_{ab}=w \text{ and } z_{ab}=i\}$.

Logbook

Carla: Worked on POS tagger (unfinished) and examined data and started on baseline first few weeks with Joost. Wrote code for computing entropy and applied this at word level and to the DOP features. Worked out the topic model and implemented the Gibbs sampling algorithm (including conditional distribution and some adjustments to the word count code that Harrie wrote for the topic model implementation). Evaluated the data (tested the annotators on the data) and wrote about information gain on word-level.

Harrie: Worked on DOP-features, got BLLIP and Disco-dop working on MAC os-x and wrote the code to create the feature-space and convert data to DOP-features. This was considerably more difficult than anticipated due to a bug in the disco-dop implementation of Andreas, however it has now been resolved in the package. Furthermore, worked out how to convert test-data to the feature space of the training-data and how to perform this task efficiently enough for the live demo. Helped Carla with the implementation of the topic model and set up the first classifier experiments on which Ties and Joost could continue.

Ties: Worked on DOP-features with Harrie. Wrote the sentence tokenizer as well as a script to run BLLIP on all data (after getting everything working). Wrote scripts to batch test classifiers for DOP-features and a general plot script to display all the statistics for DOP- and word- classifiers. Did a lot of piloting together with Joost (word features) to come up with good classifiers. During piloting, we discoverd a couple of flaws which I fixed like the splitting of our data and comparing word- and DOP- entropy. Also visualized classifier results.

Joost: Worked on POS tagger. Wrote code for the decision stumps. Implemented the baseline, tested different classifiers. Created the demo application, including the visualization, and setup the online demo. Helped debug/figure out problems with the classifiers for both the baseline and the DOP classifiers. Made small implementation improvements such as figuring out why Gibbs sampling was slow, and improved it.