University of Zurich[UZH]

Institute of Computational Linguistics

Audience Reactions in Political Speeches

A Machine Learning Approach

Mathias Müller

Contents

1 Introduction

Being persuasive is the first and foremost goal of every politician. Yet, measuring the persuasiveness of political discourse is not straightforward and I see at least a handful of ways to quantify it. You could measure persuasiveness by how often speakers are elected to the offices they have been running for, or conduct a survey in the general population to get a feeling for how a speech was received. Persuasiveness could also be related to how many positive tweets are generated by a televised speech. But an equally simple method is to count audience reactions during speeches, which is what I have settled for in this work.

Once we have settled for counting audience reactions as a way to measure persuasiveness, another problem arises. What exactly is it that makes a political speech persuasive? Since we now have a definition of persuasiveness, the question could also be phrased as: What exactly is it that generates applause or any other reaction from the audience? Obviously, an answer to this question would be immensely valuable to speakers in the political domain. There is reason to believe that, to a large extent, persuasiveness depends on the *linguistic form* a speech takes. Evidence for this claim comes from two different lines of research.

Firstly, from the point of view of traditional linguistics and the political sciences, attempts have been made to identify linguistic patterns that lead to audience reactions. Such an analysis could be performed on all levels of linguistic organisation, including vocabulary, syntactical constructs and rhetorical devices. If written text is the object of investigation, this will of course preclude some important aspects of a speech, e.g. voice quality. Studies in this spirit have merit, but they are very reminiscent of Chomskian armchair linguistics that never sees data as the primary source of information.

Therefore, secondly, computational linguistics offers a very different perspective on the problem. There, the methods are much less supervised, clearly driven by the data and additionally, many of them obviate the need for any a priori assumptions about the origin of audience reactions. For instance, a machine learning approach would not test assumptions, but rather reverse engineer the process. It would be reverse engineered in the sense that a classification algorithm would first learn to predict audience reactions. After that, one could analyse what the classifier based its decisions on and which features were most important.

In this work, I have followed the latter approach, that is, classifying audience reactions with a machine learning tool. I extracted training material from CORPS, a corpus of political speeches where audience reactions have been annotated. With this material, I conducted a series of machine learning experiments with SVM classifiers. In order to break down the problems into smaller tasks, all experiments were binary classifications. I have tested various feature combinations to see how adding features impacts classifier performance.

The remainder of this work is organised as follows. Chapter 2 reviews some of the literature both on traditional accounts of the persuasiveness of speeches in the political sciences and on their applications in natural language processing. It gives special consideration to work done on the basis of the CORPS corpus. Chapter 3 describes the CORPS corpus that serves as the source of my training material. Chapter 4 explains how I have extracted suitable training data from CORPS and the machine learning experiments I have conducted.[1] Chapter 5 presents the results of all classification experiments and also illustrates how label distributions can be exploited. Chapter 6 gives the results a thorough discussion. Finally, Chapter 7 summarises the findings and indicates future work.

[1] This chapter should leave everyone with enough information to be able to reproduce my results. If that is not so, please do not hesitate to contact me and I will gladly give more information.

1.1 Acknowledgements

I am indebted to Jasmin Stalder and Ramona Carnevale who have been working with me on this project, for discussions and their valuable comments.

2 Related Work

The idea of studying audience reactions during political speeches is not exactly new. Finding the patterns behind generating applause or similar reactions has been the holy grail to some political scientists for a rather long time. "Generating applause" is in fact the title of one such treatise by Heritage and Greatbatch (1986). Their claim is, roughly, that audience reactions are independent of the political party of the speaker and independent of their "political status" – and besides that, largely independent of the message. Instead, according to Heritage and Greatbatch, it is the *form* of the message that determines how the audience is going to react. Politicians using one of a relatively small number of rhetorical devices (e.g. *contrast*, *position taking*) clearly were more successful at generating an audience reaction.

For the task at hand – namely learning to predict audience reactions automatically – this realisation should "generate" a feeling of optimism. There is reason to be optimistic because, if the majority of audience reactions during political speeches is indeed due to the use of rhetorical devices, then those rhetorical devices can be found in written transcripts of the speeches. Moreover, the written text would then preserve all the information that is necessary to be able to predict audience reactions with machine learning. And it would be a fortunate coincidence indeed, because in most cases, the written transcript is all the information we have.

Neither is the idea of combining political sciences and computing exactly new. The political sciences have seen a significant paradigm shift towards using computationally intensive methods (see e.g. Elliott et al., 2009 for a review). A number of people have set their mind to this task before and the first step towards automatically classifying audience reactions is, of course, to create sound training data. In this regard, the most useful resource by far was published by Guerini et al. (2008): the *COrpus of tagged Political Speeches* (CORPS).

In their publication, Guerini et al. (2008) found a good balance between introducing the corpus and immediately showing what it can be used for. They investigated the relationship between the valence of expressions and the likelihood of audience reactions and then sought to compile a list of "most persuasive words" (ibid., 27). Another part of the paper is concerned with how political discourse changes after incisive historical events, where they show that the core vocabulary of George Bush's speeches changed drastically after September 11. However, machine learning was not part of their analysis.

Strapparava et al. (2010) was the first work to combine CORPS and machine learning methods and it is the most similar to my own. Their first goal was to predict "the passages in the discourses that trigger a positive audience reactions [sic]" (ibid., 1343). They extracted training instances from the CORPS corpus and defined the context of each audience reaction label as a chunk of 4 sentences before the reaction. To simplify the classification task, they reduced the number of classes by conflating all classes with positive reactions to a single class called "positive-ironical". A flavour of the SVM algorithm was used for all classification tasks.

Unfortunately, they did not explain what kind of features they used in the experiments, which makes reproducing their results impossible. But I suspect that POS features were part of the experiments because tagging is mentioned in the text. For the binary classification of "positive-ironical" against

neutral 4-sentence chunks, they report an F-measure of 0.66. In further experiments, Strapparava et al. test whether the classification performance depends on the political party of the speaker. Apparently, audience reactions are quite independent from the speaker's political affiliation – which is perfectly in line with Heritage and Greatbatch (1986) above.

On the whole, there are few publications that make use of CORPS, and surprisingly not many from outside the HLT group at the Fondazione Bruno Kessler in Trento. CORPS is an extremely useful resource and, in my opinion, it does not get the attention it deserves. The most recent publication that uses CORPS is Guerini et al. (2015), and their approach to persuasiveness is refreshingly innovative.

In a nutshell, they developed a model of the phonetic aspect of persuasiveness that they call "euphony", thus giving a phonetic score to sentences. Those scores were then used as additional features in a series of machine learning experiments. They report that a classifier trained on CORPS with phonetic score features did not outperform a classifier with basic n-gram features (ibid., 1490). However, phonetic features did help for all classifiers trained on material other than CORPS, e.g. Twitter data. But since I have chosen CORPS as the basis for my own work, phonetic scores as features are likely not worth investigating.

3 Data

For my experiments, I have extracted training material from the *CORPS* corpus (Guerini et al., 2008). CORPS is a collection of transcribed political speeches that are tagged with audience reactions. Documenting how the audience reacts to the speeches was the main focus when the corpus was built and it is therefore a good basis for my work.

The second release of the CORPS corpus (Guerini et al., 2013) contains 3600 speeches with a total of 7.9 million tokens. The majority of speakers are American politicians, which makes the corpus a somewhat unrepresentative sample. About 68.000 audience reactions are interspersed in the text. All reactions are enclosed in curly braces (`{...}`) to make sure that they can be processed automatically. Figure 1 shows examples from the corpus, where sentences were followed by a reaction from the audience. Those examples already hint at the kind of rhetoric that politicians are using to persuade their audience, a very polished and flamboyant style of speaking.

Yes, we will be one people and live the dream that will make this world free. {CHEERS}
So it turns out, lo and behold, all three U.S. automakers are now operating at a profit for the first time in years. {APPLAUSE}
And that was when I learned that he had actually, in April 2002, apparently cast a vote that would continue to allow live birth abortions in the state of Illinois. {BOOING}

Figure 1: Examples of sentences that were followed by an audience reaction, taken from CORPS

Table 1 lists all reaction labels that were used for annotation, together with their frequencies and common combinations of them[2]. The quality of annotation is very high, but there are a few instances of incorrect annotation along the lines of the example in Figure 2, taken from the file `pjbuchanan12-8-00.txt`. I have already notified the maintainers of CORPS about this and reported all inconsistencies, so the annotations are likely to improve in a future version of the corpus.

[2] For a complete description of all combinations and more corpus statistics, see the `statistics.txt` file that is distributed with the freely available CORPS corpus. It also contains statistics about all individual speakers.

class	occurrences in the corpus
{APPLAUSE}	46310
{LAUGHTER}	14055
{SUSTAINED APPLAUSE}	97
{BOOING}	756
{STANDING-OVATION}	51
{SPONTANEOUS-DEMONSTRATION}	313
{CHEERS}	234
{AUDIENCE}	1803
{AUDIENCE-MEMBER}	999
{COMMENT}	787
{OTHER-SPEAK}	404
{LAUGHTER ; APPLAUSE}	1579
{CHEERS ; APPLAUSE}	837

Table 1: Types of audience reactions and their frequencies. Very infrequent combinations of reactions are omitted here.

Most reactions in the corpus are *positive*. `{APPLAUSE}` and `{LAUGHTER}` alone make up 70 percent (50.000) of all reactions and there is only one reaction label that is undoubtedly negative: `{BOOING}`. But even in the case of `{BOOING}`, the negative sentiment is seldom against the speakers themselves. Rather, booing in this context is aimed at a common enemy, the nemesis that is threatening to take the political office the speaker is running for, or the opposite end of the political spectrum. The last sentence in Figure 1 illustrates this.

For example, he told a baffled audience in Florence, South Carolina, and I quote him directly, quote, "Rarely has the question asked, is our children learning." {LAUGHTER} Is our children learning? Well, our children is certainly not learning in Texas, governor. **CHEERS ; LAUGHTER ; APPLAUSE**

Figure 2: Inconsistent annotations in the CORPS corpus that should be enclosed by `{` and `}`

Another reason for the audiences' positive attitude could be that American politicians mostly speak within their own camp – at rallies of their party and in front of their supporters. However, the circumstances of the speeches (place and nature of the audience) are not always annotated in the corpus.

A more serious shortcoming of the reaction annotations is that there is no knowing what exactly a label corresponds to in the text. Annotators did not identify a span of text that they think caused the reaction, they simply indicated with labels the time when the audience reacted. Thus, what the audience actually reacted to might have had nothing to do with the text that was being said. In a machine learning context, this would mean that the cues that are needed to classify audience reactions cannot possibly be made available to the classifier. One would have to transform extralinguistic information into features, but the text of the speech is the only source of information that is available. In a televised speech, reactions could even be staged. For instance, the audience could be prompted to react with signs that indicate the time and manner of reaction.

All of this has an impact on the kind of research questions that can be answered with this corpus. Because of the positivity bias, any findings will not necessarily apply to audiences in general and one must be careful not to make unfounded claims about audience reactions outside of the political domain.

Because annotations do not identify spans, one cannot claim that any stretch of text "generated" the reaction – we simply don't know. Also, it cannot be assumed that the reactions of the audience represent *natural behaviour*.

Yet, keeping in mind the reservations laid out above, the CORPS corpus is a valuable source of information that can be exploited in a machine learning context. To my knowledge, it is also the only one of its kind – I do not know of any other corpus annotated in this way. If only positive or negative labels are needed, one could infer audience reactions with indirect methods. For example, by finding Twitter reactions to certain sentences said by politicians and performing sentiment analysis on them to see whether they are positive or negative. But creating training data in this way would surely be a laborious task and likely introduce greater bias than the CORPS corpus.

For some of my experiments, I have only used the `{APPLAUSE}`, `{LAUGHTER}` labels. But in scenarios where the classifier should label instances with the artificial `{REACTION}` class, all audience reactions and all their combinations are included. In Table 1, they are divided into three groups. The first group are the most straightforward audience reactions. The second set of labels are not audience reactions, strictly speaking. They are used to describe situations where other people took over the floor or intervened in some way. `{COMMENT}` labels are notes taken by the annotators. The last group are frequent label combinations.

4 Methods

This section describes the process of extracting training examples from the CORPS corpus and how training sets have been produced. Then, I explain the machine learning experiments that were conducted with the Lightside framework and the features used with the classifiers.

4.1 Extraction of training examples

I extracted training material from the CORPS corpus and prepared a series of training sets as CSV files, since this is the input format of the Lightside framework (Mayfield and Rosé, 2012). The Python script I have written to this end is called `csvify.py` and it should be available to you, together with this PDF file[3].

Figure 3 gives an overview of the extraction procedure as pseudo code, not as Python code. Each file in the corpus is read into memory, the speaker is identified and regular expressions are used to match the text of the whole speech and relevant stretches of text within it. The script then extracts spans of text from the speech, both ones that triggered a reaction and others that did not. It also extracts the latter category because not causing the audience to react can also be a classification label – I have called it the `NONE` category. All training sets that I will present later in this chapter include the `NONE` category and all experiments are set up as binary classifications of a reaction label against `NONE` .

Taking a more detailed look at Figure 3, the lines that end with "?" are optional parameters that can be controlled by the user. Here is a short description of each:

[3] I am aware that currently, changing certain parameters of the script requires editing it. If other people were to use the script, I would take care to remove global variables in favour of command line arguments and argument checking.

```
for file in corpus:
    if encoding valid:
        read all text in file
        find speaker ID
        extract speech
        extract from speech:
            tuples (sentence, reaction)
            with a certain window / one sentence before reaction?
        fix overlaps in list of tuples
        tokenize sentences
        filter by sentence length limits?
        collect neutral instances:
            tuples (sentence, NO reaction)
        filter by list of target reactions?
        conflate to single reaction type?
        balance number of instances per class?
    else: skip
save list of tuples as CSV file
```

Figure 3: Extracting training examples from the CORPS corpus (pseudo code)

- **with a certain window / one sentence before reaction?** The span of text before the reaction that is kept as its context can either be a window up to a certain length or one can use a heuristic that tries to identify exactly one sentence before the reaction.
- **filter by sentence length limits?** Only keep reaction contexts that are within the limits of the sentence length that were specified. Only applicable to single sentences before reactions, not to context windows.
- **filter by list of target reactions?** Filter out reaction labels that are not requested by the user. For instance, only keep `{APPLAUSE}` and `{LAUGHTER}`, exclude everything else.
- **conflate to single reaction type?** Map all reaction that are left at this point to a single `{REACTION}` category that is the basis for a binary classification of `REACTION` versus `NONE`.
- **balance number of instances per class?** Make sure that training examples of all classes are evenly distributed, including the `NONE` category. Unbalanced classes are likely a major confounding factor in a machine learning experiment.

It is important to emphasise the difference between enforcing a sentence length limit and defining a context window. The sentence length limit only applies when the user has requested that the script should try to determine a single sentence before a reaction. On the other hand, when a context window is defined, the span of text that is extracted as the context of the reaction will not be full sentences in most cases. However, even the context window algorithm only extracts whole words, hence always stops at word boundaries even though the window is defined in characters.

The sets that result from the procedure outlined above contain duplicates. Using a command like `uniq` on the data sets reveals that there are a number of duplicate tuples of sentences and reaction labels in them. For instance, in the largest set described below, 4500 of 110.000 instances are duplicates that appear twice, only a handful appear three times. I have decided not to eliminate them because they are not errors introduced by my script, but naturally occur in the data.

For one thing, politicians are not very inventive. They say the exact same thing on different occasions, in front of another audience of course. For instance, Dick Cheney used exactly the same sentences on October 20 and 21, 2003 (see Figure 4). Secondly, in Chapter 3 I have shown that reaction labels can be combined, resulting in e.g. `{APPLAUSE ; LAUGHTER}`. The extraction script splits up all those

combination and appends a tuple to the training set for both reactions, because, after all, the audience did react in both ways to this span of text. According to Guerini et al. (2008), there are approximately 2500 combined labels in the corpus – which results in an additional 2500 duplicates in the training set.

October 20, 2003 [...] official said, "This is the beginning of the end of America." It's pretty clear this terrorist did not know us. It's pretty clear that the terrorists who attacked us did not understand the strength and the resilience of this country. And they did not understand the determination of our President."
October 21, 2003 [...] official said, "This is the beginning of the end of America." It's pretty clear this terrorist did not know us. It's pretty clear that the terrorists who attacked us did not understand the strength and the resilience of this country. And they did not understand the determination of our President."
November 17, 2003 [...] official said, "This is the beginning of the end of America." It's pretty clear this terrorist didn't know what he was talking about. It's pretty clear the terrorists who attacked us did not understand the strength and the resilience of this country. And they did not understand the determination of our President."

Figure 4: Reuse of sentences in political speeches by Dick Cheney, referring to the aftermath of the September 11 attacks

The balancing of classes in the training sets will always be imperfect. In the current implementation, examples for the `NONE` class (that did not trigger an audience reaction) are taken from the same texts as the other reaction instances. `NONE` class examples are extracted per text file, and there are not always enough "safe" candidates that can serve as training instances. Tables 2 and 3 show how balanced the sets are, there are always more instances of a class of reactions than of the `NONE` class. If anything, this should bias the classifier towards assigning an audience reaction too often, because reactions are more prevalent than `NONE` in the training data.

4.1.1 Training sets used in the experiments

This section describes the training sets that were used in the experiments. I have created all of them with the methods described above. Table 2 shows training sets that were generated with the single-sentence context option. Table 3 shows comparable sets that only differ in that the reaction contexts are flexible windows of up to 300 characters instead of a single sentence. Both tables give the most basic statistics about the data sets, namely the labels it contains and the number of instances for each label.
The file names are to be interpreted as follows. `sc` means "sentence context" which refers to the single sentence before the reaction context. `wc` means "window context", the more flexible extraction strategy and `300` means windows of up to 300 characters are allowed. The remainder of the file names lists the reaction labels that are included in the training sets.

Regarding the various options of the extraction script, the following rules apply to all training sets:

- `{APPLAUSE}` vs. `NONE` or `{LAUGHTER}` vs. `NONE` sets are filtered to only contain reactions with their target class, applause or laughter, respectively. `REACTION` vs. `NONE` sets are *not* filtered and contain all annotation labels that are present in the CORPS corpus.
- All sets are intended to be balanced, but instances of actual reactions are not discarded just because the algorithm cannot find enough instances of the `NONE` class. Thus, positive examples are valued over negative ones.

sc-applause-none.csv		sc-laughter-none.csv		sc-reaction-none.csv	
{APPLAUSE}	39575	{LAUGHTER}	11996	{REACTION}	51571
{NONE}	34030	{NONE}	11645	{NONE}	44854
total	73605	total	23641	total	98221

Table 2: Training sets where instances have single-sentence context, i.e. the left context of a reaction label was extracted until a sentence boundary occurred

wc-300-applause-none.csv		wc-300-laughter-none.csv		wc-300-reaction-none.csv	
{APPLAUSE}	48221	{LAUGHTER}	15506	{REACTION}	65193
{NONE}	31507	{NONE}	14742	{NONE}	40424
total	79728	total	30248	total	105617

Table 3: Training sets where instances have a larger context window of up to 300 characters, regardless of sentence boundaries

- For the single-sentence context scenario, the sentence length must be between 5 and 200 characters to avoid very short or long outliers. Where it applies, a context window of up to 300 characters is allowed.
- In `REACTION` vs. `NONE` sets, labels of all kinds are *conflated* to a single label, `REACTION`. Again, those sets contain all labels that are found, not just `{APPLAUSE}` and `{LAUGHTER}`.

For one of the experiments, I have modified the sets in Table 2 to also include a CSV column with the speaker ID. Since the number of instances do not change for those sets, I have not listed them separately.

4.2 Machine learning experiments

Using the data sets introduced in section 4.1.1, I conducted a series of machine learning experiments that are described here. In all cases, I used a support vector machine (SVM; see Cortes and Vapnik, 1995) classifier that was trained on CORPS data and that labelled instances in a binary classification task.

4.2.1 Binary classification

Using Lightside (Mayfield and Rosé, 2012) as the machine learning framework, I have set up a number of experiments, all of which are binary classification tasks. Also, `{NONE}` is always one of those two labels that are predicted. I have decided to break up a multiclass labelling problem into several binary classifications because "it is often easier first to devise algorithms for distinguishing between only two classes" (Allwein et al., 2001, 113). Binary classifiers can then be combined to solve the multiclass problem, for instance by treating the binary classifiers as features for a higher-order classifier that predicts several labels. Allwein et al. also suggest ways to combine binary classifiers, which is not trivial for some algorithms, e.g. margin-based ones like SVM.

Nevertheless, I have trained SVM classifiers because, according to the Lightside user manual, SVMs are superior in many ways if the classification is only binary. Since I feel that a thorough evaluation of different classification algorithms is outside the scope of this work, I am taking this advice on faith and use SVM classifiers.

```
# applies to training sets that contain {APPLAUSE}
applause|god bless america|thank you
```

```
# applies to training sets that contain {APPLAUSE}
thanks|victory|justice|honor|future
```

```
# applies to training sets that contain {LAUGHTER}
good|now|want|funny
```

Figure 5: Examples of regex features used for the second baseline

4.2.2 Baselines

As a reference point, there are two baselines that trained classifiers can be compared to: the random baseline and a baseline that uses nothing but regex features.

First of all, the most straightforward baseline is picking a reaction class label at random. Since all classifications are binary and since the classes are almost balanced, the accuracy of the random baseline will be around 0.50. In other scenarios, there could be a need to pinpoint the random baseline more precisely, for instance because the classification is only slightly better than the random baseline – and one would still need to know if a classifier performs above chance. To improve the random baseline, one could either balance the classes very strictly or implement a solution that picks labels at random. Strapparava et al. (2010) also had random choice as their baseline.

The second baseline consists of using *only* the regular expression features of the Lightside interface. The regex features simply list words and phrases that can occur in the reaction contexts. If the regex matches the text, the feature is either set to 1 or the regex hit counter is incremented (if counting the hits is requested). I myself have decided on the words and phrases in the regex features, so this baseline has a quite supervised method. I have closely looked at the data before that, and the set of words is also informed by the list of "most persuasive words" in Guerini et al. (2008, 27). Figure 5 gives examples of regex features that were used for the second baseline.

4.2.3 Features

In order to beat the baselines described in section 4.2.2, I have trained SVM classifiers with the following features. For each type of feature, I also give a rationale for why I think it will be useful.

- **unigrams:** some words in the reaction context will be indicative of how the audience reacted to what was said, if at all
- **bigrams:** frequent combinations of words have more discriminatory power than unigrams. For instance, "thank you" could be a bigram feature
- **skip stopwords:** stopwords occur frequently in all contexts and are usually ignored in machine learning contexts
- **n-gram normalisation and occurrences count:** enhancements to the unigram and bigram features. Repetitions of words or phrases could have an impact on audience reaction, and normalisation is used because the line lengths are variable
- **line length:** the audience might be more likely to respond to shorter sentences, because they can be understood more quickly. There is no point in reacting if the speaker has already moved on
- **speaker ID:** some speakers might be more persuasive or funny than others. So, the information on who is speaking might be important

Towards the end of the experiments, it became clear that audience reactions might also have to do with the *time frame* of a speech. The audience is expected to react at the very beginning and end of the speech, or when speakers take turns. This could be captured in a feature like the sentence number, counted from the beginning of the speech. However, this feature is not part of my experiments.

4.2.4 Experiments

I have trained a number of classifiers with various feature combinations. In order to gauge the influence of individual features, I have taken care to add them to classifiers individually.

For each of the `sc` training sets, a basic classifier was trained that only used unigram features. On top of them, I added features incrementally to see if they improve the results. The features that were added subsequently are bigrams, skip stopwords, n-gram normalisation and n-gram occurrence counts. Features were always extracted with a rarity threshold of 5.

This was followed by experiments that test whether the line length or speaker ID make a difference for the classification. Thus, in addition to the features mentioned so far, line length and the ID of the speaker were included. The line length feature only makes sense in the single sentence condition (`sc`), and the speaker ID feature was also only tested with `sc` sets.

Finally, I investigated the reaction context, i.e. whether 1) trying to find one sentence before the reaction or 2) a window up to certain amount of characters before a reaction would lead to better results. Therefore, all steps above are repeated for the `wc` training sets, except for adding the line length and speaker ID features. It is worthwhile to juxtapose these two methods of extracting training examples because the decision to only extract a single sentence before a reaction as its context is completely unfounded and arbitrary.

All classifiers trained in this way are common soft margin SVMs with the default settings in Lightside. Automatic cross-validation was performed on the data, which meant 10-fold cross validation in all cases because of the size of the training material. As the primary outcome of these experiments, I have computed the accuracy of the classifiers.

5 Results

Overall, the results show that all binary classification tasks perform well above chance and well above the the regex baseline, too. They also reveal that in the single-sentence tasks, adding features is in most cases detrimental to the performance. The inverse is true for the tasks where the training material had a context window of 300 characters.

5.1 Classification tasks

Table 4 shows the performance of classifiers where the training material only had instances with single-sentence context (`sc`). The column headers indicate the training set that was used for each column and their names must be interpreted as follows: `sc` denotes the kind of context as explained above. `applause`, `laughter`, `reaction` and `none` are labels present in the training sets. Consequently, `sc-applause-none` means that the classifier trained with this material had little context for each reaction

		sc-applause-none	sc-laughter-none	sc-reaction-none
BL	random baseline	0.5000	0.5000	0.5000
	regex baseline	0.5377	0.5308	0.5433
Features	unigram	0.7087	**0.7834**	0.7023
	+bigram	**0.7103**	0.7763	**0.7090**
	+skip stopwords	0.7014	0.7601	0.7024
	+n-gram normalisation	0.7014	0.7601	0.7024
	+count occurrences	0.7038	0.7639	0.7008
	+line length	0.7048	0.7634	0.7009
	+speaker ID	0.7034	0.7618	0.6985

Table 4: Accuracies of SVM classifiers trained for different classification tasks (columns) and various feature combinations (rows). Best values for each task are set in **bold**. BL = Baselines, sc = single sentence context, {applause, laughter, reaction, none} = classification labels.

and that the classification labels where `{APPLAUSE}` against `{NONE}`.[4]

In general, adding features to the classifiers in Table 4 degraded their performance, albeit not by much. Adding the bigram feature was the only modification that improved accuracies slightly, but only when classifying `{APPLAUSE}` vs `{NONE}` and `{REACTION}` vs `{NONE}`. Skipping stopwords always decreased the performance and adding the speaker ID as a feature never led to an improvement either. N-gram normalisation did not have any detectable effect on the overall accuracy. Counting n-gram occurrences and the line length feature slightly improved the results in 2 out of 3 cases.

Changing the single-sentence context to a flexible character-based context window presents a similar picture, but adding features does no longer have adverse effects. Table 5 shows those results. Initially, the classifiers in Table 5 were trained with the exact same method as the first set of classifiers in Table 4, but the outcome was surprising. Accuracies of those classifiers were always above 0.99 - which of course meant that the training method had a serious shortcoming. After investigating the data, it turned out that virtually all instances of the `{NONE}` class had punctuation characters at the end – while examples of the other classes never had a punctuation character at the end. Therefore, I excluded punctuation characters from the n-gram features.

With this small modification in place, the performance of classifiers whose training material had window context is similar to the single-sentence context ones and the results do not get better. But now, adding features either led to improvements or did not change accuracy at all. Skipping stop words and n-gram normalisation did not have any impact on the performance.

Taken together, the results from both tables indicate that a `{APPLAUSE}` vs `{NONE}` binary classifier takes the right decision in 7 out of 10 cases. `{LAUGHTER}` vs `{NONE}` classification is even more reliable, and the classifier's decision will be correct in 4 out of 5 cases.

5.2 Identifying highest-scoring sentences

Besides training classifiers, I have also closely inspected the label distributions of the resulting classifiers. It occurred to me that once the label distribution and the feature weights are known, it would be interesting to try and generate the most typical sentence for a certain label. Put another way, I wanted

[4] Only using regex features sometimes causes Lightside to behave in unexpected ways, and the GUI often displays all instances being labelled with the same class. I have reported this to the Lightside user group, but did not receive a satisfactory answer so far.

		wc-300-applause-none	wc-300-laughter-none	wc-300-reaction-none
BL	random baseline	0.5000	0.5000	0.5000
	regex baseline	0.6021	0.5335	0.6219
Features	unigram	0.6905	0.7648	0.7053
	+bigram	0.6762	0.7821	0.6948
	+skip stopwords	0.6909	**0.7865**	0.7069
	+n-gram normalisation	0.6909	**0.7865**	0.7069
	+count occurrences	**0.6934**	0.7517	**0.7099**

Table 5: Accuracies of SVM classifiers trained for different classification tasks (columns) and various feature combinations (rows). Best values for each task are set in **bold**. BL = Baselines, wc-300 = window context, {applause, laughter, reaction, none} = classification labels.

to generate the one sentence the classifier would be the most confident in labelling with a certain class and that would get the highest score. Ideally, such a sentence would get a score of 1.0 from the classifier for one class, and 0.0 for the other.

Eventually, I have decided that instead of generating the most typical sentence for a given class label, I would simply *find* the most typical sentences in the training data. That is, the ones that received the highest scores from the classifier. To this end, I have exported the label distribution data of the best classifier for `{APPLAUSE}` vs `{NONE}` and `{LAUGHTER}` vs `{NONE}` (see Tables 4 and 5). Then, I have analysed the distributions with another Python script that extracts the top n sentences for each class label form the CSV data. The script is called `top_instances.py` and should be available to you as a supplement to this document.

Some examples of highest-scoring training instances are given in Table 6[5]. The first thing to notice is that there *are* instances in the data where the classifier is very confident indeed about making a decision. In the case of `{LAUGHTER}` some of the text snippets even reach a perfect score of 1.0. Interestingly, the top instances labelled with `{APPLAUSE}` are much longer than the top instances of `{LAUGHTER}`. This could mean that, if one were to set up a classification scenario where both of them have to be classified, the line length would be a good discriminator between the classes.

6 Discussion

Despite their simplicity, n-gram features are already very powerful and hard to beat. This is a commonplace realisation in machine learning. Very often, sophisticated features and exotic feature combinations fail to outperform n-gram features. The results in Table 4 confirm this, although it could be argued that skipping stopwords, normalisation and counting occurrences also count as n-gram features. Still, both the line length and the speaker ID (arguably not n-gram features) feature fail to improve the system.

The speaker ID feature failing to improve the system could be interpreted as follows. Obviously, it is not a personal trait of certain politicians to be funnier than others. At least, this trait is not as consistent as to let the identity of a speaker inform the classifier of whether their sentence has triggered an audience reaction. Alternatively, it could mean that the audience does not react more often, even if a politician is particularly funny. This hypothesis could be tested by counting the number of reactions

[5] It much more insightful, of course, to run `top_instances.py` on an exported label distribution yourself and look at more than a handful of examples.

{APPLAUSE} score	{NONE} score	text
0.999999996	0.0000000032	George Stevens and all the commission members who are here will you please stand and let us give you a round of applause
0.999999991	0.0000000085	We will have when the Congress passes this year 's budget three years of deficit reduction in a row for the first time since Harry Truman was President of the United States of America
0.999999938	0.0000000613	We will work to help end lawsuit abuse because we know that it 's a lot easier for America 's businesses to hire new workers if they don't have to hire lawyers

{LAUGHTER} score	{NONE} score	text
1.0000000000	0.0000000000	As all of you know I 'm in a family business too
1.0000000000	0.0000000000	That means that in the White House women are in charge of everything abroad and everything at home
1.0000000000	0.0000000000	Fortunately she was also involved in the business
1.0000000000	0.0000000000	Yes I like a guy who follows in his father 's footsteps
1.0000000000	0.0000000000	I 'm kind of warming up for the State of the Union

Table 6: Upper part: instances from the training data that got the highest `{APPLAUSE}` score. Lower part: instances from the training data that have a "perfect" `{LAUGHTER}` score

per speech and normalise the values – or simply aggregate the *tag density* values in the CORPS statistics file, which are very similar.

Keeping track of the line length did not help either, which means that it is not the length of an utterance which is crucial for the audience to react in time. That is, understanding what was said and reacting with e.g. laughter before the politician moves on. Although the line length is not a good discriminator in the binary classification experiments I have conducted, it might still be useful to discriminate between `{APPLAUSE}` and `{LAUGHTER}` (see Section 5.2). In this work, there is no classification scenario where both of them have to be classified as separate labels.

Moving on to the second set of experiments reported in Table 5, extracting more textual context for reactions did not increase accuracies. It follows from this observation that more context does not solve the classification problem. Put another way, it is not that the single-sentence context classifier simply lacked information which was there, just farther away from the reaction in the text. As I have already pointed out in Chapter 5, adding features to the context window classifiers does no longer degrade performance. This might be the case because the usefulness of the features that are added incrementally increases with the size of the text sample.

Despite all the reservations voiced above, it seems to me that, given the difficulty of automatically processing humour and similar pragmatic aspects of written texts, the accuracies of approximately 0.7 and 0.8 are surprisingly good. They can be taken to mean that in the majority of cases, a politician evokes a reaction from their audience with means that are actually present in the written text. The remaining 20 to 30 percent of reactons might indeed be due to extralinguistic factors that are not captured in the data. Besides, the results are in line with Strapparava et al. (2010) who reported an f-measure of 0.66 for a very similar task (see Chapter 2).

7 Conclusion

This work describes methods to extract training data from the CORPS corpus. Using a Python script, I have produced a number of data sets extracted from CORPS. The data sets were then used as training material in machine learning experiments with Lightside. All tasks were binary classifications and started out with simple baselines. Taking the baselines as the point of departure, I then incrementally added features to the classifiers and calculated their accuracies. None of the features added to the classifiers could bring significant improvement. Additionally, I have analysed the label distributions to reveal the instances that get the highest score from the classifier.

If there would be a continuation of this work, I would suggest that it be about one of the following topics:

- experimenting further with the context window to find an optimal context size. After all, members of the audience have at their disposal a much larger context at any point in time.
- adding as a feature the relative *position* of a sentence in the text, to account for highly ritualised situations during speeches, where the audience is bound to react. For instance, the very beginning and end of speeches would likely be easy to detect with such a feature.
- adding as a feature POS tagging, or combinations of words and their POS tag, to reduce the sparseness of data. Lightside comes with a built-in POS tagger and predefined options for POS features.
- adding more classification scenarios, e.g. `{APPLAUSE}` vs. `{LAUGHTER}`, but also consider multi-class tasks
- collecting more diverse training material. Currently, the speeches in CORPS are almost exclusively given by American politicians, which introduces a heavy bias.

References

Allwein, E. L., Schapire, R. E., and Singer, Y. (2001). Reducing multiclass to binary: A unifying approach for margin classifiers. *The Journal of Machine Learning Research*, 1:113–141.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Elliott, E. W., Ho, K., and Holmes, J. S. (2009). Political science computing: A review of trends in computer evolution and political science research. *Journal of Information Technology & Politics*, 6(2):166–175.

Guerini, M., Giampiccolo, D., Moretti, G., Sprugnoli, R., and Strapparava, C. (2013). The new release of corps: A corpus of political speeches annotated with audience reactions. In *Multimodal Communication in Political Speech. Shaping Minds and Social Action*, pages 86–98. Springer.

Guerini, M., Özbal, G., and Strapparava, C. (2015). Echoes of persuasion: The effect of euphony in persuasive communication. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1483–1493, Denver, Colorado. Association for Computational Linguistics.

Guerini, M., Strapparava, C., and Stock, O. (2008). Corps: A corpus of tagged political speeches for persuasive communication processing. *Journal of Information Technology & Politics*, 5(1):19–32.

Heritage, J. and Greatbatch, D. (1986). Generating applause: A study of rhetoric and response at party political conferences. *American Journal of Sociology*, pages 110–157.

Mayfield, E. and Rosé, C. (2012). Lightside: text mining and machine learning user's manual.

Strapparava, C., Guerini, M., and Stock, O. (2010). Predicting persuasiveness in political discourses. In Calzolari, N., Choukri, K., Maegaard, B., Mariani, J., Odijk, J., Piperidis, S., Rosner, M., and Tapias, D., editors, *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, Valletta, Malta. European Language Resources Association (ELRA).