

# AgendaMiner

Machine learning tools for analyzing local government agendas

**Jeffrey Barrera**

June 1st, 2016

A thesis submitted to the Graduate Program in Journalism at Stanford University in partial fulfillment of the requirements for the degree of Master of Arts in the Department of Communication.

Advised by Dan Nguyen

Copyright © 2016

Jeffrey Barrera

ALL RIGHTS RESERVED

# Abstract

The agendas of local government meetings provide a useful way to monitor civic proceedings without needing to attend dozens of public meetings. However, reading agendas by hand is still a time-consuming process, especially since only a small fraction of the proceedings are likely to be of interest to most watchdogs. Therefore, I explore using machine learning techniques to predict which items on an agenda are likely to be worthy of further investigation. I first describe an approach to converting raw agendas in several formats into structured JSON files, and then test a variety of algorithms for classifying the agenda items. I find that a combination model can correctly locate 83% of items that were hand-classified as interesting in a validation dataset, though with a low precision rate (17%) that means there will be many false positives.

The code for this project is available at [github.com/AgendaMiner](https://github.com/AgendaMiner).

# Acknowledgements

Two years ago, I could not have undertaken this project. As an undergraduate Urban Studies major, my coursework focused much more heavily on public policy issues and qualitative research than on advanced statistics and computational methods. That I'm now able to discuss the trade-offs of support vector machines and cross-validated logistic regression is thanks to a fantastic group of Stanford professors and doctoral students.

In the Political Science department, Clayton Nall and Simon Ejdemyr introduced me to data science, Andy Hall drilled home Monte Carlo methods and probability distributions, and Justin Grimmer and Bobbie Macdonald provided a foundation in machine learning principles and working with text as data. In the Computational & Mathematical Engineering department, Gabriel Maher and Alexander Ioannidis explained more advanced machine learning approaches like boosting and bagging. And in the Journalism program, Cheryl Phillips and Dan Nguyen taught me how to scrape websites and PDFs, clean up messy data, and use computational approaches to shed light on civic issues. Dan also served as my advisor for this thesis, guiding me through the process of writing 2,855 lines of code.

Jacob Fenton, Mindy Huang, and Jeremy Singer-Vine all provided additional technical help. Jeremy's PDFPlumber library greatly simplified the process of extracting

formatting information from PDFs, and he was kind enough to quickly update the library when I encountered some PDFs it couldn't handle. Jacob provided excellent advice on dealing with PDFs, and was an awesome partner for projects in Prof. Grimmer's class (thanks for putting up with my down-to-the-wire scrambles!). Mindy's suggestions proved invaluable throughout the project, including helping me devise the approach to structuring PDFs.

Finally, Louise Auerhahn at Working Partnerships USA graciously provided me with several years worth of agenda-monitoring reports, which formed the basis of the training data for this project.

# Contents

<b>Local Governments: Outside the Spotlight.....</b>	<b>1</b>
What About Agendas?.....	1
Let's Automate This .....	2
<b>On the Agenda .....</b>	<b>5</b>
<b>Parsing the Agendas.....</b>	<b>6</b>
HTML Agendas.....	8
PDF Agendas .....	9
<b>Classifying Agenda Items .....</b>	<b>17</b>
Creating a Set of Interesting Items.....	17
Building a Model .....	22
<i>Logistic Regression .....</i>	<i>23</i>
<i>Support Vector Classifier .....</i>	<i>23</i>
<i>Naive Bayes .....</i>	<i>23</i>
<i>K-Nearest Neighbors.....</i>	<i>24</i>
<i>Predicting Topics .....</i>	<i>24</i>
<i>Combined Models .....</i>	<i>25</i>
<i>Any-of-the-Above.....</i>	<i>26</i>
<b>Discussion &amp; Future Steps.....</b>	<b>28</b>
<b>Appendix .....</b>	<b>32</b>
<b>References.....</b>	<b>35</b>

# Local Governments: Outside the Spotlight

Local governments have a huge impact on our daily lives. They shape our options for getting to work, the condition of our schools, and how far away the nearest ambulance will be in an emergency. Yet these agencies typically receive far less media coverage and public scrutiny than their state and federal counterparts. While dedicated press bureaus cover statehouses, Congress, and the White House, there may not be a single reporter at many transportation commission, school board, and county supervisor meetings.

This gap in coverage is understandable. For a news organization with limited resources, it makes more sense to send a reporter to the state Capitol — whose decisions will affect everyone in the state — than to try to attend dozens of meetings at cities, counties, and special districts. But as a result, the vast majority of government decisions are made outside the watch of the Fourth Estate.

## What About Agendas?

Meeting agendas provide a way to help bridge this gap. In California and many other states, local governments are required to post agendas listing the topics that will be discussed before each meeting.<sup>1</sup> These agendas provide a way to track what actions

---

<sup>1</sup> This is a mandate of the Ralph M. Brown Act, California Government Code § 54954.2(a).

are being taken without going to every meeting, and to decide which meetings could be worth attending. And since they are published in advance, they can give reporters time to research a topic and interview sources before the meeting occurs — especially important since background knowledge and an understanding of the jargon is often necessary to follow the proceedings.

Still, trying to read and understand every agenda item can be overly time-consuming for a busy reporter covering a number of agencies. In California's Santa Clara County, for example, there are 13 cities, 2 towns, 36 school and community college districts, and 28 special districts (public entities that oversee water systems, hospitals, public transit, and the like).<sup>2</sup> Since most agencies meet multiple times a month, and each meeting can cover 20 to 50 or more items, there could easily be over 5,000 agenda items a journalist would have to sift through every month in search of the few things that might be newsworthy.

## **Let's Automate This**

Reading that many items can be daunting for a human, but what about for a computer? Filtering through agendas for potentially interesting items has several attributes that make it seem suitable for machine learning techniques. First, the primary

---

<sup>2</sup> List compiled from the [Santa Clara County Local Agency Formation Commission](#), the [Association of Bay Area Governments](#), the [Santa Clara County Office of Education](#), and the [California Community Colleges Chancellor's Office](#).

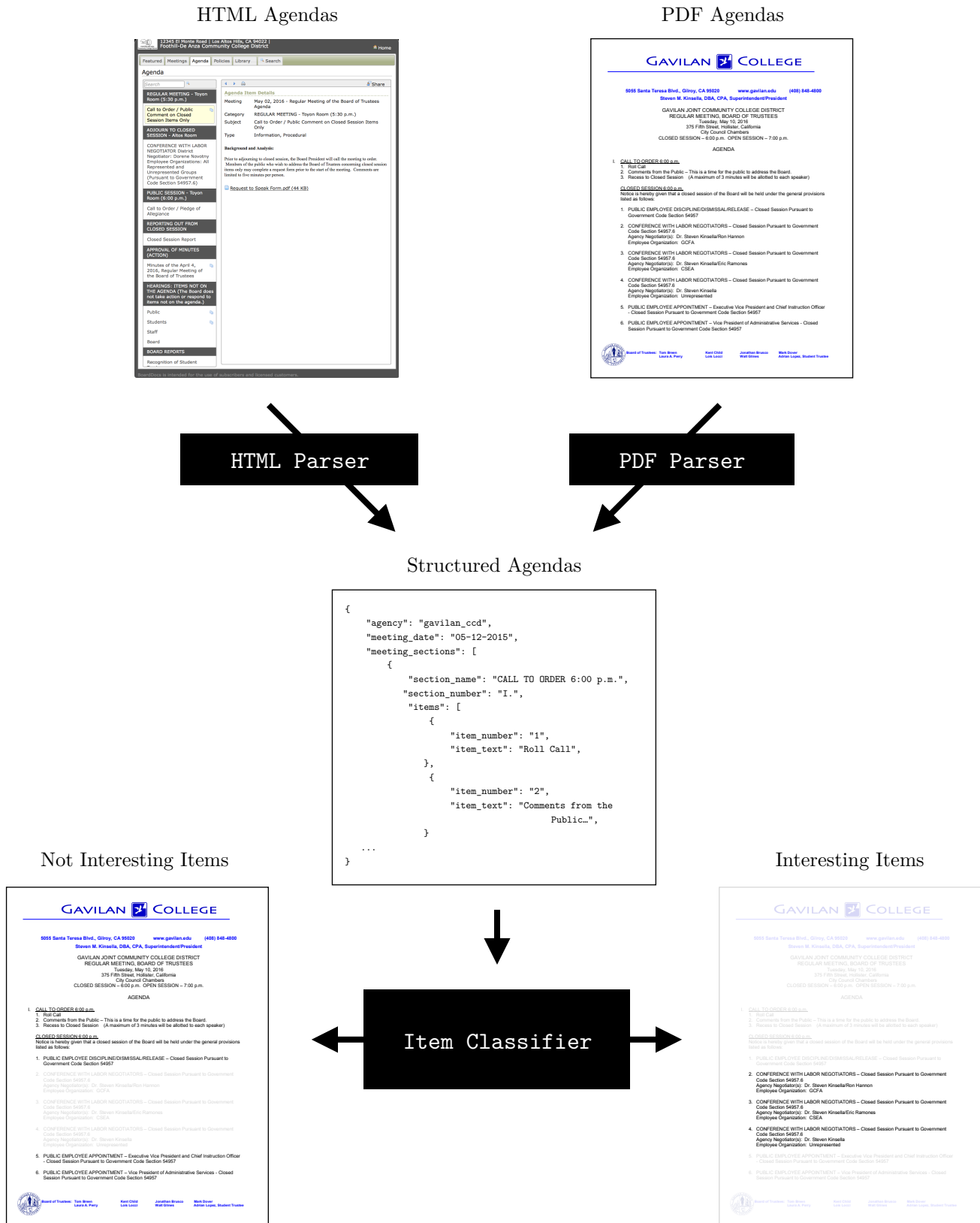


reason this task is challenging to do by hand — the sheer number of items — means there are enough data points for an algorithm to find meaningful patterns. Second, most agenda items fall into a limited number of topics that come up repeatedly (approving contracts, negotiating with labor unions, listening to comments from the public, and so on). Third, agendas tend to use a relatively consistent grammar of terms and phrases: the jargon that can make meetings hard to follow as an outsider has the upside that the same words will typically be used whenever a topic comes up for discussion.

Given these promising attributes, I decided to explore how parts of this agenda-monitoring process could be automated with machine learning approaches. In this paper, I describe my two-stage process (visualized in FIGURE 1) for analyzing the agendas — first converting each agenda into a standardized format that a computer could understand, and then training machine learning algorithms to predict whether items were likely to be interesting. I then evaluate how well this approach performed, and discuss avenues for future research.

All the code for this project was written in Python, and can be found at [github.com/AgendaMiner](https://github.com/AgendaMiner). The code to structure agendas is in the `agenda-parser` repository, while the item-classification code is in `item-classifier`.

FIGURE 1: THE AGENDA-ANALYSIS PROCESS

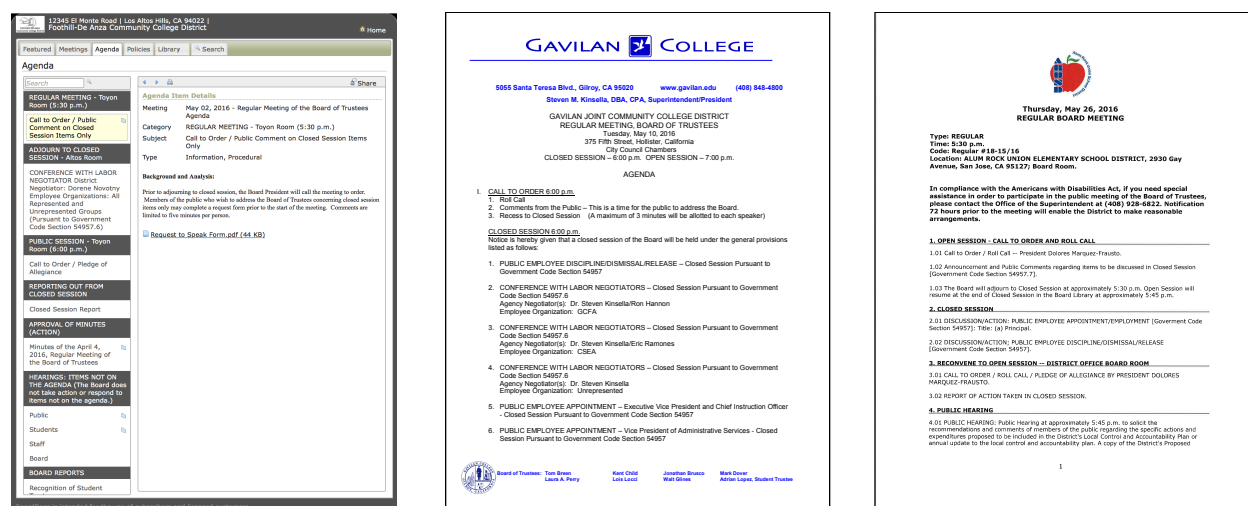


# On the Agenda

First, let's see what these agendas actually look like. As the examples in FIGURE 2 show, the formatting of agendas varies depending on the agency and the process used to create the agenda. Some agencies put them online as HTML, others provide a PDF, and some print out a paper document, scan it, and then upload the scanned version. Still, the underlying structure is more-or-less the same across every agency:

- The meeting is broken up into a series of **sections**. These vary from agency to agency, but often include a closed session (not open to the public) where the board discusses confidential matters like employee discipline, a consent calendar of routine items (approving minutes, paying bills), a public comment time when anyone can

FIGURE 2: SAMPLE MEETING AGENDAS



An HTML agenda from Foothill-De Anza Community College District, a text-based PDF from Gavilan Community College District, and a scanned image-based PDF from Alum Rock Union School District (see the Appendix for larger versions of these agendas).

speak to the board, a set of informational items that don't require a vote, and a set of action items that the board does vote on.

- Each section contains one or more **agenda items**. These are the specific presentations that the board will listen to, proposals the board will vote on, and so on. Typically the agenda includes just a short description of each item, while a larger “board packet” will contain staff memos and other materials with more information on each item.

While the full board packet can tell you more about each item, the description on the agenda is often enough to decide whether an item is worth further investigation or not. Moreover, the full packets are far more complicated for a computer to deal with — they are often provided as scanned images since they contain letters and other documents the agency received in hard-copy, and the formats of the materials in them vary widely. Therefore, I focused on the agendas for this project.

## Parsing the Agendas

Before I could start analyzing the agendas, I needed a way to convert them into a standard, machine-readable structure. This meant dealing with the three main formats that were used by the various government agencies: HTML, text-based documents that were saved as PDFs, and PDFs that were just scanned images of printed pages. Because

of the extra challenges involved in dealing with image-based PDFs (such as having to use finicky and imperfect Optical Character Recognition tools to extract the text from the images), and because agencies thankfully seem to be moving away from providing agendas in this format, I chose to ignore it and just work on HTML and text-based PDFs.

For both formats, my goal was to output a uniform structure as a JSON (JavaScript Object Notation) file. JSON is a simple format that can describe hierarchical content (such as meeting sections that contain agenda items) in a way that both humans and computers can understand. FIGURE 3 gives an example of what a JSON-formatted agenda looks like.

FIGURE 3: SAMPLE JSON-FORMATTED AGENDA

---

```
{
  "agency": "gavilan_ccd",
  "meeting_date": "05-12-2015",
  "meeting_sections": [
    {
      "section_name": "CALL TO ORDER 6:00 p.m.",
      "section_number": "I.",
      "items": [
        {
          "item_number": "1",
          "item_text": "Roll Call",
        },
        {
          "item_number": "2",
          "item_text": "Comments from the Public - This is a time for the
public to address the Board",
        }
      ]
    },
    ...
  ]
}
```

## HTML Agendas

An increasing number of agencies have begun using agenda-management platforms like BoardDocs, IQM2, and Agenda Online, which produce HTML-formatted agendas. Since HTML is a markup language that (at least when used semantically, which not all platforms do) provides some structure, it's relatively straightforward to write a parser that finds the section headings and agenda items. Unfortunately, each platform requires a custom parser, but at least that code then covers all the agencies on the platform.

Due to time limitations, I just wrote one parser for the BoardDocs platform as a proof of concept. This parser is actually two scripts, both inside the **agenda-parser** repository:

1. **scrapers/board\_docs\_scraper.py**: This uses the Requests library to download a list of all the agendas for a given agency, and then uses BeautifulSoup to extract the titles, meeting dates, and links to each agenda. This information is saved off into an agenda-list.json file for each agency.
2. **parsers/board\_docs\_parser.py**: This takes the agenda-list.json file and downloads all the items on each agenda. It then organizes the items by meeting section, pulls out information like the item number and item type (informational, action, procedural, etc), and writes the structured agenda to a JSON file.

Together, this process is able to convert the agendas to my standardized JSON format with practically-perfect accuracy. I ran the scripts on three different agencies that use BoardDocs, and successfully converted all 586 agendas that the scraper found.

## PDF Agendas

Handling PDFs was decidedly more challenging. Unlike HTML, the PDF format doesn't provide any semantic information about the structure of a document. Instead of saying that Line A is a headline and Line B is a paragraph, it simply says that the character located two inches from the top of the page and one inch from the left edge of the page is in bold, 18-point Times New Roman. Moreover, practically every agency uses their own unique formatting. One agency may put section headings in bold type and put numbers before agenda items, while another uses roman numerals for sections and sets items in a different typeface. To handle this variety and lack of structure, I decided to write a generalized parser that uses machine learning to teach the computer to read the PDFs like a human would. This involved a four-step process:

1. **Extracting the text and formatting from the PDF:** Using the PDFPlumber library, I pulled out each line of text in an agenda by finding all the characters that were the same distance from the top of the page (plus or minus a small error margin), and ordering them by their distance from the left edge of the page. To avoid boilerplate text (the name and address of the agency, for example), I filtered


out any text that was offset from the main content with a line or was inside a box. I then created a set of features that described the formatting of each line, such as the font it was in, the font size, whether it was bold or italic, if it started with a number, how much it was indented from the left edge of the page, if it was all capitalized, and so on (see FIGURE 4 for a visual example of some of these features). I then saved out a CSV (Excel-like spreadsheet format) file with one row for each line of text and columns for all the features. This code can be found in `agenda-parser/parsers/pdf_parser.py`.

2. **Creating a set of training lines:** I then asked a human user (myself, though this step doesn't require any technical skill) to manually classify the lines in a few sample agendas from each agency. My script (also in `pdf_parser.py`, with the `manual_classify` flag set to true) presented the user with one line at a time, and asked them to assign it to one of four categories:

1. **Section Heading:** A line that described the start of a new section.
2. **Item Heading:** The first line of text of each agenda item.
3. **Item Text:** Additional lines of text for an agenda item. This was given its own category since the first line often began with a number or other special formatting that the following lines didn't have.



FIGURE 4: SAMPLE FORMATTING FEATURES



**CUPERTINO UNION SCHOOL DISTRICT**

**Board of Education**  
**Regular Meeting**

**AGENDA**

Cupertino Union School District  
10301 Vista Drive  
Cupertino, CA 95014

Tuesday, March 26, 2013  
6:30 p.m.

Welcome to the meeting of the Cupertino Union School District Board of Education. If you would like to address the Board during Public Comments on any agenda item or any item not on the agenda, please fill out a comment card available in the hallway and give it to the Administrative Assistant. You will be called on to comment during this time and comments will be limited to three (3) minutes. To ensure that all speakers are provided an equal opportunity to address the Board during Public Comments, individual speakers may not "yield" their allotted time to address the Board to other speakers. In addition, the Board may, in accordance with the Brown Act (section 54954.3[b] of the Government Code), limit the total amount of time allocated for comment on a particular issue. The Board may choose to respond to agenda item comments or reserve their responses for discussion and action when the agenda item appears during the course of the meeting. It should be noted that the Board discourages complaints against individual officers or employees of the District during open session.

*Individuals who require special accommodation should contact the Superintendent's Office at (408) 252-3000, ext. 200 at least two business days before the meeting date.*

**As a courtesy to others, please turn off your cell phone upon entering the meeting.**

**1. CALL TO ORDER / FLAG SALUTE** ← Font: Arial, Font-style: bold, Font-size: 14pt, All-caps: true  
Inset: 50

**2. COMMUNICATION – EMPLOYEE ORGANIZATIONS**

**3. PUBLIC COMMENTS** ← Font: Arial, Font-style: regular, Font-size: 10pt, All-caps: false  
Inset: 60

The public may address the Board on any agenda item and any item not on the agenda. The Board president will only call on those who have filled out comment cards before the meeting, and each speaker will be allotted three (3) minutes. The Board will not respond or take action on any non-agenda item comments at this time, although the item may be agendized at a later date (Ed. Code 35145.5). The comments shall be made from the podium. In accordance with Government Code 54954.2 – No action or discussion shall be undertaken on any item not appearing on the posted agenda. The Board shall limit the total time to 30 minutes.

**4. CONSENT ITEMS**

4.1 Award of Hazardous Material Inspection and Testing Services in the amount of \$25,275 to Sensible Environmental Solutions (SES) for Data Infrastructure Upgrade Projects at Dilworth, Garden Gate, McAuliffe, Meyerholz, Regnart and Stevens Creek Campuses ← Font: Arial, Font-style: regular, Font-size: 12pt, All-caps: false  
Inset: 70

4.2 Award of Roofing Inspection Services for 25 Campuses in the Amount of \$123,680 to Allana Buick and Bers

**5. SUPERINTENDENT'S REPORT**

**Boilerplate:**  
**Ignored**

*Identified by the box & the knowledge that this is the first page*

Starts with a number

Starts with a sub-number

4. **Other Text:** Irrelevant lines that weren't part of a section heading or item, such as boilerplate that hadn't been ignored earlier.

The script then saved out a CSV like the one created in step 1, but with additional columns indicating which category each line had been assigned to.

3. **Classifying the lines:** I next fed the hand-classified lines from step 2 into a machine learning algorithm (from the scikit-learn library) that tried to work out which features are associated with each category for a given agency. To give the algorithm more information to work with, I also created a document-term matrix using the text of each line (see sidebar for more information). To explore which algorithm would work best, I randomly split the hand-classified lines into two groups: a training set containing two-thirds of the lines, and a validation set with the remaining third. For each algorithm, I then built a model with the training set, and checked how well it performed on the testing set. I tried logistic regression with a cross-validated L1 penalty, a Support Vector Machine (SVM), and a random forest. The SVM model was mediocre, but both the logistic regression and the random forest worked extremely well, correctly classifying over 95% of the lines in the validation set. Ultimately, I settled on using the random forest, since — as shown by the example tree in FIGURE 5 — it can capture more complicated feature relationships than a logistic model (Breiman 2001). I then generated a random forest model for

## SIDEBAR: DOCUMENT-TERM MATRICES

---

Also known as a “bag of words” approach, document-term matrices are a common strategy for feeding text into algorithms that can only work with numeric data (Grimmer 2016). They involve creating a new feature for each word in any of the documents, with its value set to the number of times the word was used in a given document (in this case, in each line of text). For example, the phrase “the cat in the hat” could be encoded as:

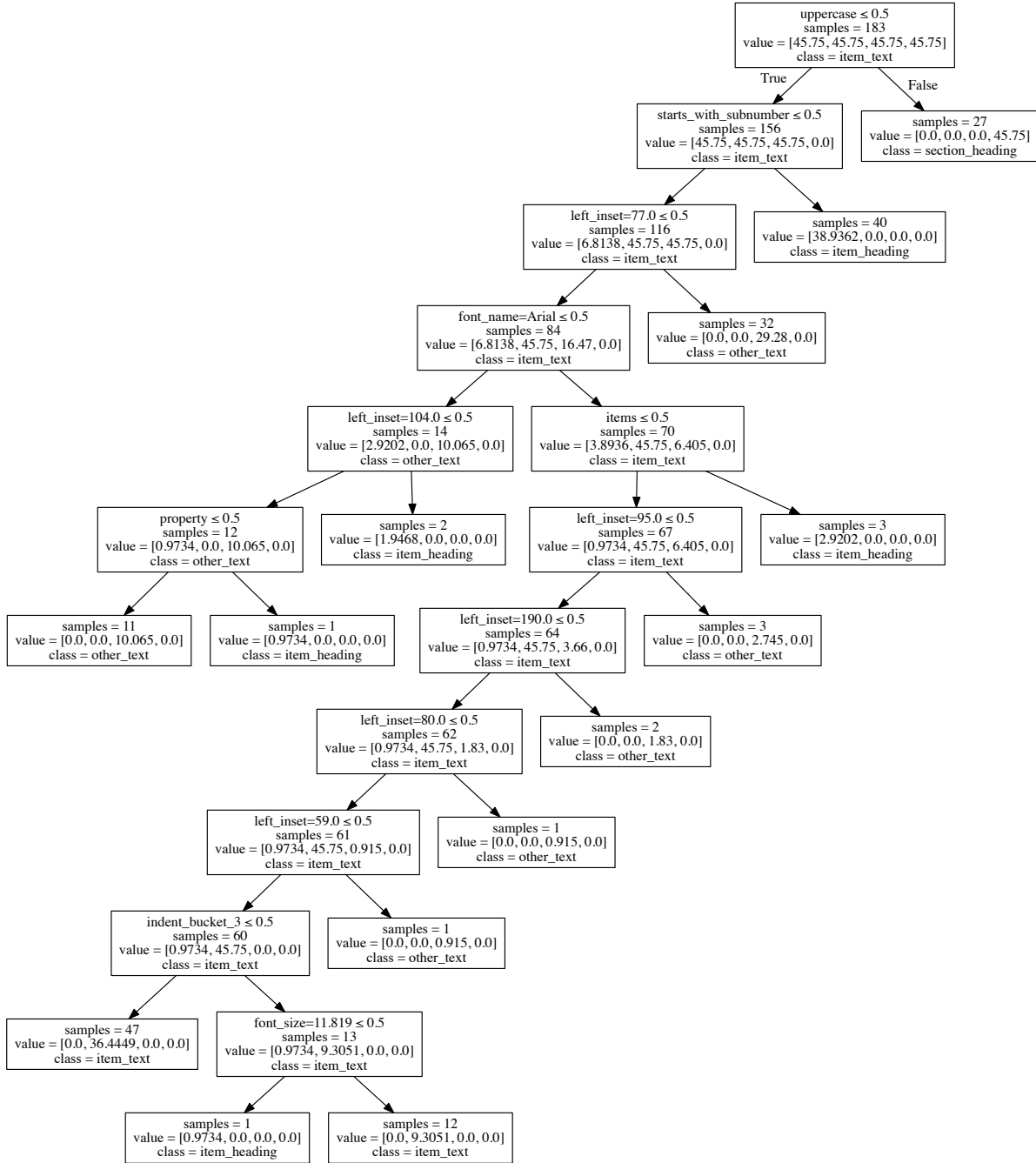
"the"	"cat"	"in"	"hat"
2	1	1	1

While this approach is simple, it ignores the order in which words appear, so this approach couldn’t tell the difference between “the cat in the hat” and “hat in the cat,” even though one phrase is rather more concerning than the other. To get around this, we can create features for each pair of words (known as a bigram). That would look like:

"the cat"	"cat in"	"in the"	"the hat"
1	1	1	1

For this project, I used single words, bigrams, and trigrams (sets of three words). I also excluded overly-common words like “the” and “and,” since these occur too frequently to provide helpful information.

FIGURE 5: SAMPLE CLASSIFICATION TREE



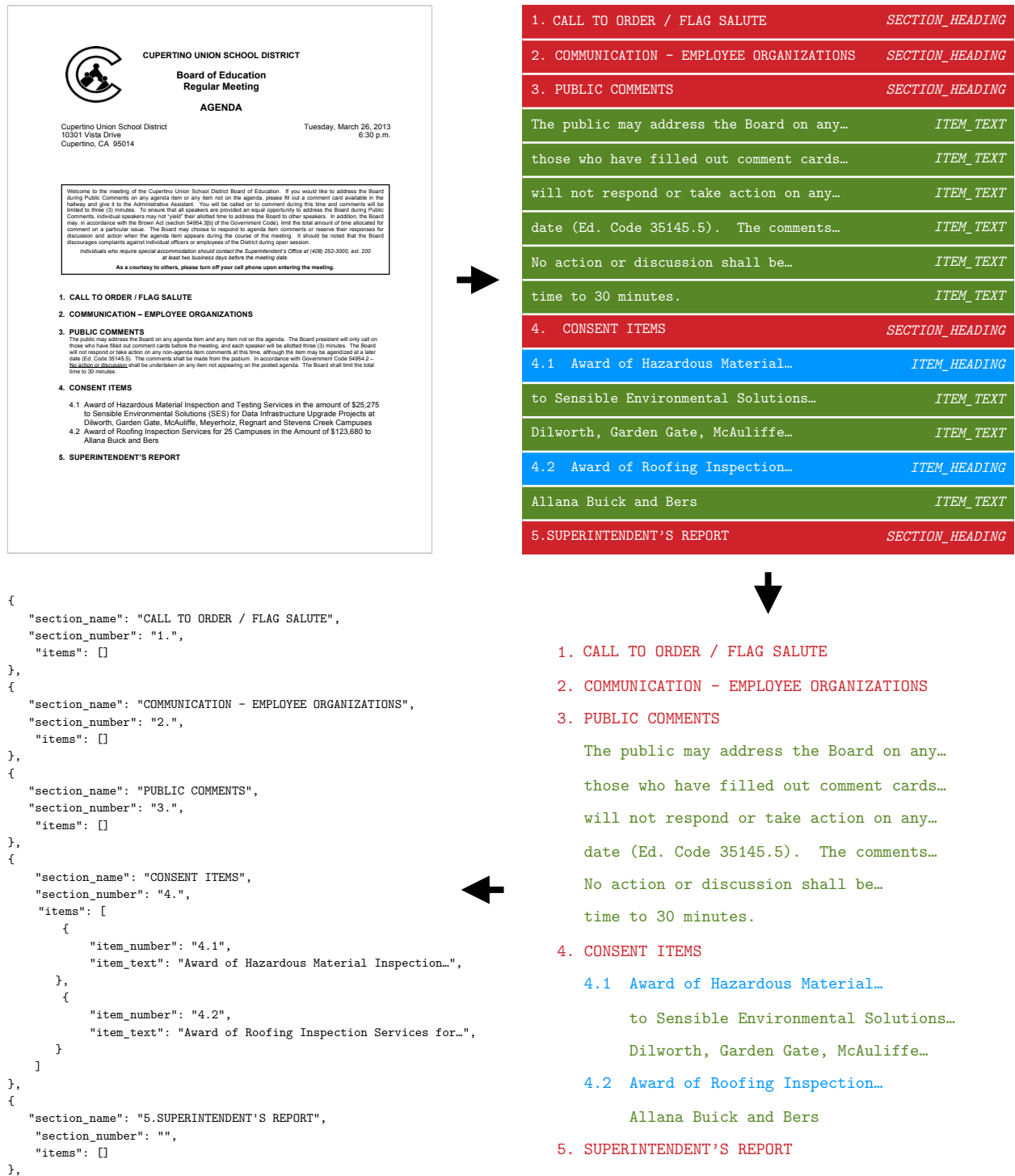
A sample classification tree, created by splitting on features that best separate different observations in the dataset. The first line in each box lists the feature that is being split on, and the bottom line gives the resulting classification. The random forest creates many such trees, each fitted on a different sample of the data, and then takes the averages of all the trees.

each agency’s PDFs, and used it to predict a category for every line of text. This code can be found in `agenda-parser/parsers/line_classifier.py`.

4. **Structuring the lines:** Finally, I used the predicted categories from step 3 to convert the lines into structured data. I demarcated sections by the lines categorized as section headings, and then assigned the lines categorized as items to the most recently listed section (see FIGURE 6 for a depiction of this process). I then saved the agenda to a JSON file with the same structure as that used for HTML agendas. This code can be found in `agenda-parser/parsers/line_structurer.py`.

For the most part, this process proved quite effective at turning raw PDFs into structured agendas. I applied it to 84 agendas from two different agencies, and then randomly spot-checked the resulting JSON against the original PDFs. With just a couple of exceptions, practically all the section headings were successfully matched, as were the first lines of almost all the agenda items. However, the approach struggled with finding items that deviated from the standard formatting (sometimes agendas would use a different style for closed session items than they did for everything else, for example). Additionally, at times the models struggled to distinguish between lines categorized as “item text” and “other text,” since these often had similar formatting. As a result, if the text of an agenda item spanned multiple lines, some of the lines might be dropped.

FIGURE 6: LINE STRUCTURING PROCESS



The PDF structuring process, clockwise from top left: (1) lines & formatting features are extracted from the PDF, (2) each line is assigned a line-type (indicated by color), (3) the lines are structured using the line-type classifications, and (4) the result is saved as JSON.

Both these bugs stem from what I think is the biggest weakness in this formatting-based approach: the model has no knowledge of the context surrounding a given line. The line classifier can't treat closed session items differently, or know that a line following an item heading is more likely to be item text, since that structure won't be inferred until the next step of the process. Therefore, I think the best way to further improve this parser is to add a second pass. After the classified lines are structured in step 4, this approach would add a new set of contextual features to each line (for example, what is the category of the previous and next lines? Is this line inside a section? Which section is it inside?). Each line would then be re-classified by a model incorporating these additional features, and then re-structured with the new classifications. Unfortunately I didn't have time to implement this second pass as part of this thesis project, but I hope to in the future.

## **Classifying Agenda Items**

### **Creating a Set of Interesting Items**

Once I had all the agendas in a standardized format, I could work on predicting which agenda items were likely to be noteworthy. To do this, however, I needed a set of items that had been hand-classified as worthy of further investigation that I could use as training data. For this, I reached out to Working Partnerships USA, a public policy

research and advocacy group based in San Jose, CA. To help track what local governments were working on, and so they knew if there were upcoming topics they should pay attention to, the Working Partnerships research staff has been manually monitoring the agendas of many agencies in Santa Clara County.<sup>3</sup> They kindly agreed to give me several years worth of their weekly reports, which listed all the items the research team considered important.

While the reports covered several dozen local governments, I focused on just three agencies for the item matching process: the San Jose Evergreen Community College District, the Foothill-De Anza Community College District, and Eastside Union High School District. This decision was partly due to timing — when I began working on the item classifier, my PDF structuring pipeline wasn't yet robust enough to provide high-quality JSON output, and these three agencies all used the BoardDocs platform that I had a working parser for. I also wanted similar agencies to simplify the model-building process — the topics that are important at a school district are likely quite different from those that matter at cities or transit agencies.<sup>4</sup>

---

<sup>3</sup> I had previously interned with Working Partnerships, which helped give me the idea for this project in the first place.

<sup>4</sup> That said, it should be fairly easy to generalize this in the future by either combining multiple models, or adding features indicating the type of agency and interacting these with the other features in the model.



To build my training set, I needed to match the descriptions in the reports with the agenda items in the JSON files. I initially thought this would be fairly straightforward, since the reports listed the name, date, agency, and number of each item. I therefore wrote a parser (`agenda-parser/parsers/docx_parser.py`) to extract this information from the Microsoft Word-formatted reports, filtered to just the three agencies for which I'd parsed all the agendas, and then created another script (`item-classifier/report_item_matcher.py`) that found corresponding item for each entry in the reports.<sup>5</sup>

However, this was only able to match about 60% of the items listed in the report. I found that since the reports were intended for a human audience, many entries combined several items into a single report listing, used a slightly different name, or had other quirks that my matching script couldn't handle. After attempts to make the matcher more robust still couldn't catch a significant number of items, I ended up hand-matching the remainder.<sup>6</sup> At the end of this process, I had 253 items that were classified as positive (interesting), and 8,347 items covering the same timespan that were not classified as interesting.

---

<sup>5</sup> I also combined and flattened the JSON files into a Pandas DataFrame with one row for each item, to make it easier to feed the data into a scikit-learn algorithm in the next step.

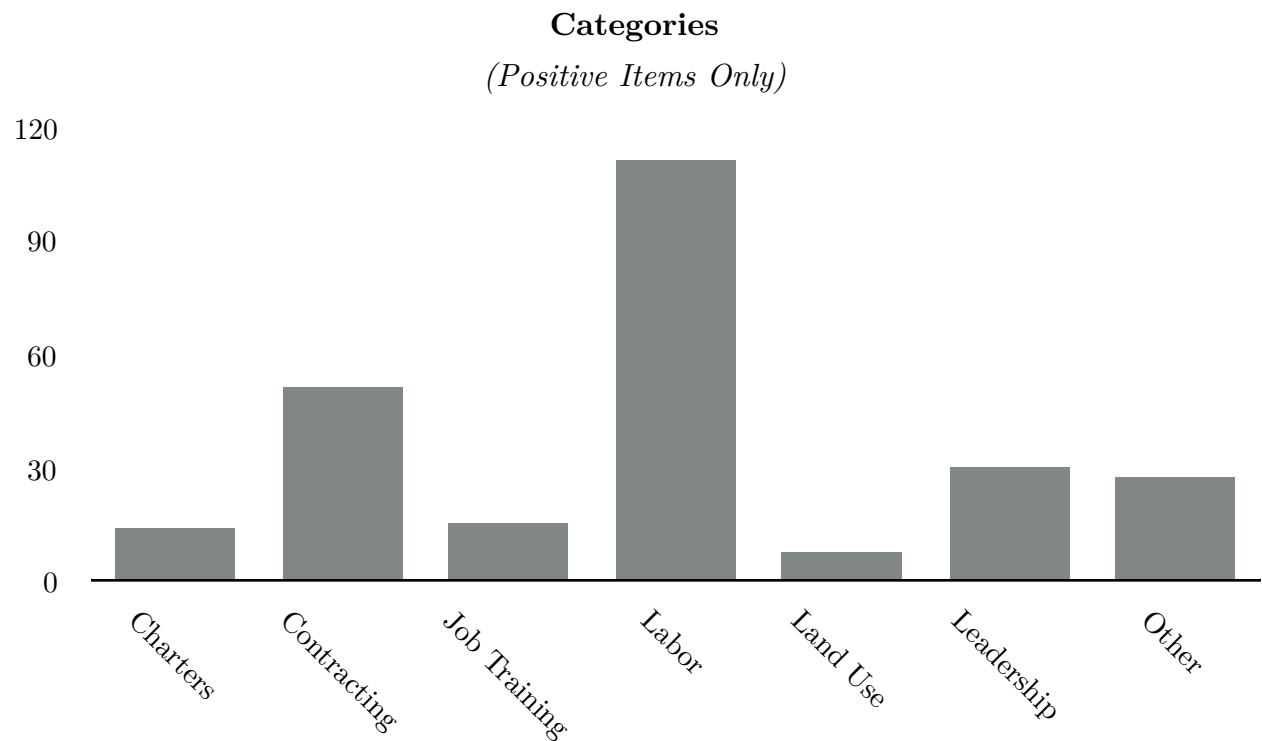
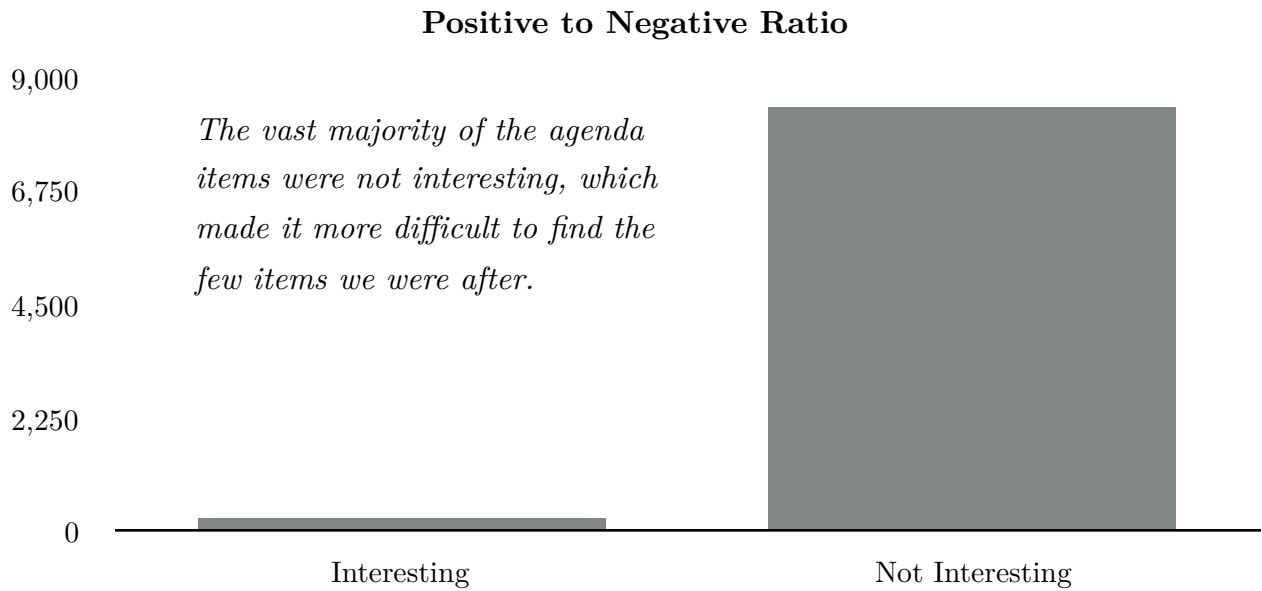
<sup>6</sup> Needing to hand match some items to ensure a clean dataset unfortunately also meant that I couldn't easily incorporate additional agencies once I got the PDF parser working better — by this stage I simply lacked the time to do the additional hand-matching that would have been necessary.

To help train and evaluate the classifiers I would build in the next step, I also hand-categorized the 253 positive items into one of seven topics:

1. **Charters:** Items about charter schools.
2. **Contracting:** Items that related to the agency awarding contracts to outside companies.
3. **Job Training:** Items about career and technical education projects, such as internships, trade apprenticeships, and the like.
4. **Labor:** Items about wages, job standards, and negotiations with labor unions.
5. **Land Use:** Items about the physical property the agency owned, such as selling land.
6. **Leadership:** Items that affected how the agency was run, such as performance evaluations of superintendents, changes to election rules, and appointments to board and commissions.
7. **Other:** A miscellaneous classification for items that didn't fit into one of the other categories.

FIGURE 7 shows how many items fell into each category.

FIGURE 7: AGENDA ITEM BREAKDOWN



*Conferencing with labor negotiators is a frequent closed-session item, which helps explain why that category is so large.*

## Building a Model

With my set of positive items prepared, I was ready to build my prediction model. Since I was trying to predict text using algorithms that can only handle numbers, my feature set was a document-term matrix similar to the one I used in the PDF parser, again generating unigrams, bigrams, and trigrams from the words in each item. I also again randomly split my dataset into two groups, a training set with two-thirds of the items, and a validation set with the remaining third.<sup>7</sup> I then tried using a variety of algorithms to build models from the training data, and tested how well each performed on the validation data.

For each algorithm, I looked at two evaluation metrics. I was primarily interested in the *recall* rate: the percent of the items that were actually classified as interesting that the model thought were interesting. However, since it's possible to achieve a perfect recall rate by simply assuming every item is interesting, I also looked at the *precision* of each model — how many of the items it classified as positive actually were interesting.<sup>8</sup> Here's how the models compared:<sup>9</sup>

---

<sup>7</sup> To ensure there were enough of the comparatively few positive observations in each group, I stratified the two sets so each contained an equal proportion of interesting items.

<sup>8</sup> Since the training and validation sets were randomly split every time the models were generated, these metrics fluctuated by between two and twelve percentage points on each run. To control for this, the numbers I report are the averages across five runs.

<sup>9</sup> All of these were implemented using the scikit-learn library, and the code can be found in `item-classifier/item_classifier.py`.

### *Logistic Regression*

I started by using logistic regression with a cross-validated L1 penalty. This penalty helped regularize the model and prevent overfitting, by ignoring features that didn't significantly help the model. The baseline set by this model wasn't great: it had a 40% recall rate, though its precision (60%) was quite high. Basically, this model was quite conservative — it only found a fraction of the items, but at least these tended to actually be interesting.

### *Support Vector Classifier*

Next, I tried a support vector machine (SVM) classifier with an RBF kernel, using a grid search to set the gamma and C hyperparameters. This generated a much more flexible model that could capture more complicated relationships between the features (James et al 2013). When applied to the validation set, it had 63% recall and 30% precision.

### *Naive Bayes*

Naive Bayes is a popular approach for dealing with text data, applying Bayes' theorem to the terms to predict the probability an observation belongs to a given category (Jurafsky 2016). This model had the highest recall — 81% — of any individual

algorithm. It also had the lowest precision (17%), though as I'll discuss later this is less concerning that it would be in some other applications.

### *K-Nearest Neighbors*

I then tried a K-nearest neighbors model with K set to 3. This is a non-parametric approach that simply compares each item in the validation set to the three most similar items in the training set, and assign the new item to the most frequently occurring class among those three (Altman 1992). I hoped this might catch more unusual cases missed by the other model-based techniques, but unfortunately it only had 31% recall (though with 75% precision). I'm guessing that because such a small proportion of the observations were positive, any signal that could have been provided by this approach was drowned out by the noise of all the negative items.

### *Predicting Topics*

At this point, I wondered if part of the reason the models were having trouble was because I was asking them to find any interesting item, when in fact I knew there were several different topics within that group. Therefore, I next tried to see how well I could predict the seven categories into which I'd hand-classified the data. I tested both a cross-validated logistic regression and a naive Bayes model. The regression performed poorly on nearly every category, with at best 56% recall on items classified as labor. As

shown in TABLE 1, the Bayesian model did better, finding 82% of the labor items, 60% of the charter school items, 52% of job training items, 49% of contracting items, and surprisingly, 26% of the items in the miscellaneous category (which I'd expected to be the hardest to predict correctly). However, it failed to find nearly all of the land use and leadership items.

TABLE 1: TOPIC CLASSIFICATION PERFORMANCE

	Logistic Regression		Naive Bayes	
	<i>Precision</i>	<i>Recall</i>	<i>Precision</i>	<i>Recall</i>
<b>Charter Schools</b>	85%	28%	16%	60%
<b>Contracting</b>	34%	11%	7%	49%
<b>Job Training</b>	80%	16%	14%	52%
<b>Labor</b>	81%	56%	22%	82%
<b>Land Use</b>	0%	0%	0%	0%
<b>Leadership</b>	48%	14%	6%	16%
<b>Other</b>	0%	0%	4%	26%

### *Combined Models*

With mixed results from these individual models, I moved on to exploring if I could do better by combining the models. I took the predicted probability that an item was positive from each original model, along with the predicted topics from the topic classifiers, and then used these predictions as features for a combined model. I tried three different algorithms for generating this combined model:

1. **Logistic Regression:** With 40% recall and 64% precision, this approach performed only slightly better than the original logistic regression classifier.
2. **SVM Classifier:** This model had 49% recall and 50% precision, putting it in the middle of the pack.
3. **Naive Bayes:** This did the best of the combined models, with 66% recall and 34% precision. However, this wasn't much better than the original SVM classifier, and had lower recall than the original naive Bayes classifier (albeit with higher precision).

In short, none of the combined models performed notably better than the original classifiers that used n-grams as features. I'm guessing that for these combined models to work well, they would need additional contextual information about the item, so they could know to give more credence to the SVM in some instances, while favoring naive Bayes in others. I had hoped the predicted topics from the topic classifiers would provide this context, but it seems they didn't work well enough to provide meaningful data. In the future, I'd like to try providing this context by first using a clustering algorithm like K-Means to group similar items together, and then using the cluster assignments as a feature in a combined model.

### *Any-of-the-Above*

Finally, to see just how high I could push the recall rate, I tried one more approach: treating an item as interesting if any of my four original models classified it

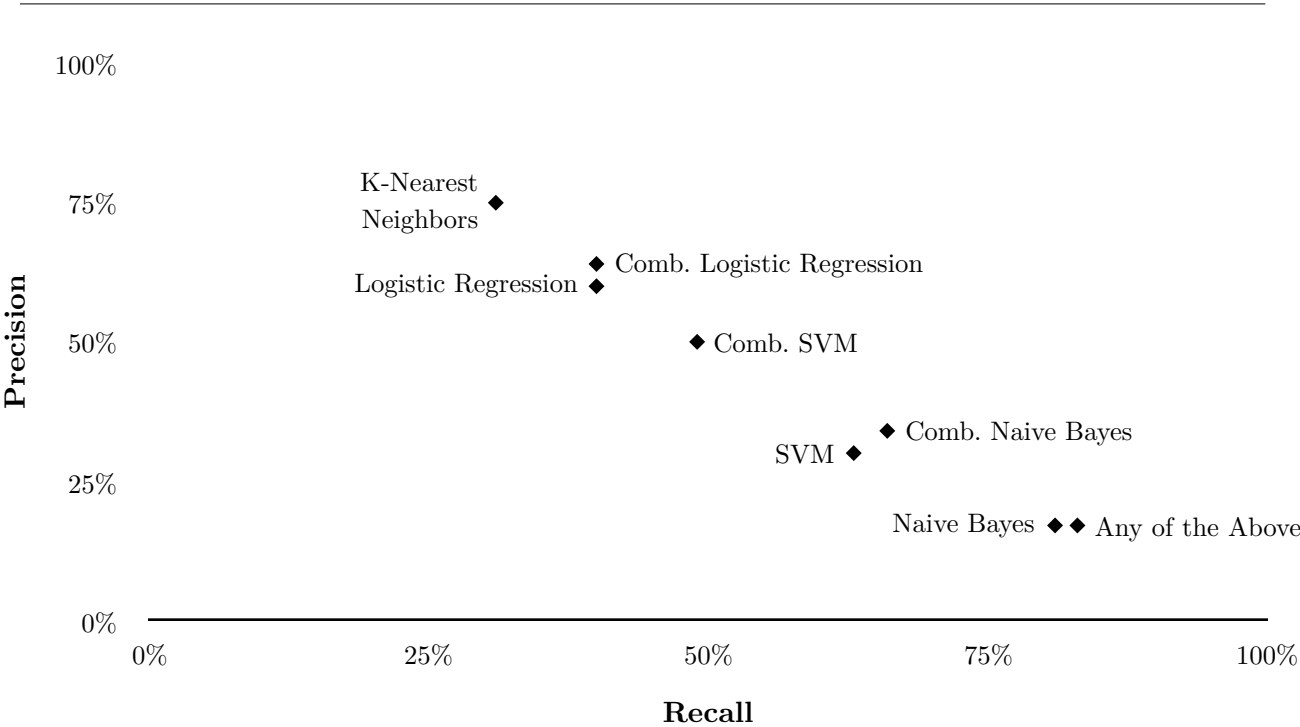


positively, or if the naive Bayes topic classifier assigned it to the labor, charter, or job training categories (these were the three topics where that model had better than 50% recall). As TABLE 2 and FIGURE 8 show, this any-of-the-above model produced the highest recall rate — 83% — of any approach. I expected its precision to drop considerably, but it actually ended up with the same (albeit low) precision as the original naive Bayes classifier, at 17%.

TABLE 2: MODEL PERFORMANCE COMPARISON

	Logistic Regression	SVM	Naive Bayes	K-Nearest Neighbors	Comb. Logistic Regression	Comb. Naive Bayes	Comb. SVM	Any of the Above
<b>Recall</b>	40%	63%	81%	31%	40%	66%	49%	83%
<b>Precision</b>	60%	30%	17%	75%	64%	34%	50%	17%

FIGURE 8: THE PRECISION/RECALL TRADEOFF



## Discussion & Future Steps

So what do these performance metrics say about the real-world applicability of using machine learning to classify agenda items? After all, 83% recall seems decent, but that still lets a fair number of potentially noteworthy items slip through undetected. And getting recall that high comes at the cost of low precision, meaning a human is still going to need to sort through a number of false positives.

While the answer is somewhat dependent on the use-case, on the whole I think this approach holds significant potential. For a journalist who currently doesn't have the time to read through 5,000 agenda items a month, using the any-of-the-above approach can cut out 85% of the items, leaving 750 items to review. That's a somewhat more reasonable number, though most of these will likely still not be newsworthy. Since the journalist will likely have other sources that can alert them to especially newsworthy events, it may be worth using a more conservative models (perhaps one of the SVMs) that would miss more items, but return fewer false positives. The combined SVM, for example, would give the reporter just 74 items a month to read, the majority of which should be potentially interesting.

For a group like Working Partnerships, where it's important to catch every single item of interest, 83% recall is not good enough to remove the need for human review. However, this model could provide a first pass over the items, simplifying the manual

review process by flagging most of the interesting items. And with some additional work, it's likely possible to train an algorithm to take an opposite approach — finding items that are almost certainly *not interesting*, and removing or de-emphasizing them to save the human reviewer time.

The obvious next stage for both these use-cases is to study ways to keep recall high while reducing the number of false positives. Beyond the potential improvements already mentioned through this paper, there are two other main avenues I'd like to explore in the future:

1. **Incorporating the full board packet:** As I mentioned earlier, the full board packet contains more information about each item, which could provide a far richer dataset for these algorithms to work with. Since many of these packets are scanned PDFs, however, I would have to use an OCR tool like Tesseract to extract the text from each document, and then would need to match supporting documents to their corresponding agenda item. Both of these steps are far more time-consuming than I was able to fit into this thesis project, but could be extremely useful — the agenda listing may only mention that there is a potential construction project, while an attached memo might discuss funding complications or community opposition, for example. Incorporating this text could thus improve a model's ability to distinguish between interesting and uninteresting items with similar descriptions on the agenda.

## 2. Using Natural Language Processing to extract more meaningful features:

Right now, I'm simply using a "bag of words" approach by converting the text of the items into a document-term matrix. While this approach works surprisingly well, it's still rather crude — simply relying on word counts rather than any understanding of what a sentence actually means. Using more advanced tools like the Natural Language Toolkit or Google's TensorFlow, it may be possible to identify more specific information, such as the amount and bidders on a contract, which staffers or board members are associated with an item, and so on. This kind of knowledge could also improve a classifier's ability to distinguish between items — a \$500,000 contract for repairs to a school site might be more routine than spending \$500,000 on a marketing agency, for example.

Even without these enhancements, though, I think there are already several valuable outcomes to this project. First, a pipeline for converting agendas into structured data opens the door to a variety of potential applications, from web applications that aggregate multiple agendas to various kinds of analysis (for example, feeding the agendas into a topic-modeling algorithm to explore what subjects different agencies focus on). Second, I think the PDF parser — especially if I implement the second-pass step that should further improve its accuracy — provides a useful template

for extracting information from other kinds of documents that rely on visual cues to suggest their structure.

Finally, I hope that showing machine learning models can find useful patterns in public documents can help validate future efforts to shed light on government proceedings by statistically analyzing textual data. As the traditional models for funding watchdog journalism slip away, I believe computational methods are increasingly important to cost-efficiently providing civic accountability. I therefore hope this project can play some small part in contributing to the development of these methods.

# Appendix

Here are larger examples of the sample agendas from FIGURE 2.

## HTML AGENDA: BOARDDOCS

The screenshot displays the BoardDocs agenda for the Foothill-De Anza Community College District. The header includes the district's name and address: 12345 El Monte Road | Los Altos Hills, CA 94022 | Foothill-De Anza Community College District. A navigation bar contains links for Featured, Meetings, Agenda, Policies, Library, and Search. The main content area is titled 'Agenda' and features a search bar and a 'Share' button. The agenda items are listed in a sidebar on the left, including 'REGULAR MEETING - Toyon Room (5:30 p.m.)', 'Call to Order / Public Comment on Closed Session Items Only', 'ADJOURN TO CLOSED SESSION - Altos Room', 'CONFERENCE WITH LABOR NEGOTIATOR', 'PUBLIC SESSION - Toyon Room (6:00 p.m.)', 'Call to Order / Pledge of Allegiance', 'REPORTING OUT FROM CLOSED SESSION', 'APPROVAL OF MINUTES (ACTION)', 'HEARINGS: ITEMS NOT ON THE AGENDA', 'BOARD REPORTS', and 'APPROVAL OF CONSENT CALENDAR (ACTION)'. The main content area on the right displays 'Agenda Item Details' for the 'May 02, 2016 - Regular Meeting of the Board of Trustees Agenda'. It includes fields for Meeting, Category, Subject, and Type, followed by a 'Background and Analysis' section with a paragraph of text and a link to 'Request to Speak Form.pdf (44 KB)'. The footer contains copyright information for BoardDocs by Emerald Data Solutions, © 2002-2016.

12345 El Monte Road | Los Altos Hills, CA 94022 | Foothill-De Anza Community College District

Home

Featured Meetings Agenda Policies Library Search

### Agenda

Search

REGULAR MEETING - Toyon Room (5:30 p.m.)

Call to Order / Public Comment on Closed Session Items Only

ADJOURN TO CLOSED SESSION - Altos Room

CONFERENCE WITH LABOR NEGOTIATOR  
NEGOTIATOR District Negotiator:  
Dorene Novotny Employee  
Organizations: All Represented and  
Unrepresented Groups (Pursuant to  
Government Code Section 54957.6)

PUBLIC SESSION - Toyon Room (6:00 p.m.)

Call to Order / Pledge of Allegiance

REPORTING OUT FROM CLOSED SESSION

Closed Session Report

APPROVAL OF MINUTES (ACTION)

Minutes of the April 4, 2016, Regular Meeting of the Board of Trustees

HEARINGS: ITEMS NOT ON THE AGENDA (The Board does not take action or respond to items not on the agenda.)

Public

Students

Staff

Board

BOARD REPORTS

Recognition of Student Trustees

California Community College Trustees Board Report

APPROVAL OF CONSENT CALENDAR (ACTION)

Approval of Consent Calendar

1. Ratification of Contracts and Agreements

2. Ratification of Board Warrants Greater than \$5,000 for February 2016 and March 2016

### Agenda Item Details

Meeting May 02, 2016 - Regular Meeting of the Board of Trustees Agenda

Category REGULAR MEETING - Toyon Room (5:30 p.m.)

Subject Call to Order / Public Comment on Closed Session Items Only

Type Information, Procedural

**Background and Analysis:**

Prior to adjourning to closed session, the Board President will call the meeting to order. Members of the public who wish to address the Board of Trustees concerning closed session items only may complete a request form prior to the start of the meeting. Comments are limited to five minutes per person.

[Request to Speak Form.pdf \(44 KB\)](#)

BoardDocs is intended for the use of subscribers and licensed customers. All users are required to read and follow the acceptable use policy.

BoardDocs® by Emerald Data Solutions, © 2002-2016.

An HTML agenda from Foothill-De Anza Community College District, available at <http://www.boarddocs.com/ca/fhda/Board.nsf/Public>.



5055 Santa Teresa Blvd., Gilroy, CA 95020      [www.gavilan.edu](http://www.gavilan.edu)      (408) 848-4800  
Steven M. Kinsella, DBA, CPA, Superintendent/President

GAVILAN JOINT COMMUNITY COLLEGE DISTRICT  
REGULAR MEETING, BOARD OF TRUSTEES  
Tuesday, May 10, 2016  
375 Fifth Street, Hollister, California  
City Council Chambers  
CLOSED SESSION – 6:00 p.m. OPEN SESSION – 7:00 p.m.

AGENDA

- I. CALL TO ORDER 6:00 p.m.
1. Roll Call
  2. Comments from the Public – This is a time for the public to address the Board.
  3. Recess to Closed Session (A maximum of 3 minutes will be allotted to each speaker)

CLOSED SESSION 6:00 p.m.

Notice is hereby given that a closed session of the Board will be held under the general provisions listed as follows:

1. PUBLIC EMPLOYEE DISCIPLINE/DISMISSAL/RELEASE – Closed Session Pursuant to Government Code Section 54957
2. CONFERENCE WITH LABOR NEGOTIATORS – Closed Session Pursuant to Government Code Section 54957.6  
Agency Negotiator(s): Dr. Steven Kinsella/Ron Hannon  
Employee Organization: GCFA
3. CONFERENCE WITH LABOR NEGOTIATORS – Closed Session Pursuant to Government Code Section 54957.6  
Agency Negotiator(s): Dr. Steven Kinsella/Eric Ramones  
Employee Organization: CSEA
4. CONFERENCE WITH LABOR NEGOTIATORS – Closed Session Pursuant to Government Code Section 54957.6  
Agency Negotiator(s): Dr. Steven Kinsella  
Employee Organization: Unrepresented
5. PUBLIC EMPLOYEE APPOINTMENT – Executive Vice President and Chief Instruction Officer - Closed Session Pursuant to Government Code Section 54957
6. PUBLIC EMPLOYEE APPOINTMENT – Vice President of Administrative Services - Closed Session Pursuant to Government Code Section 54957



Board of Trustees: Tom Breen  
Laura A. Perry

Kent Child  
Lois Locci

Jonathan Brusco  
Walt Glines

Mark Dover  
Adrian Lopez, Student Trustee



**Thursday, May 26, 2016  
REGULAR BOARD MEETING**

**Type: REGULAR**

**Time: 5:30 p.m.**

**Code: Regular #18-15/16**

**Location: ALUM ROCK UNION ELEMENTARY SCHOOL DISTRICT, 2930 Gay Avenue, San Jose, CA 95127; Board Room.**

**In compliance with the Americans with Disabilities Act, if you need special assistance in order to participate in the public meeting of the Board of Trustees, please contact the Office of the Superintendent at (408) 928-6822. Notification 72 hours prior to the meeting will enable the District to make reasonable arrangements.**

**1. OPEN SESSION - CALL TO ORDER AND ROLL CALL**

1.01 Call to Order / Roll Call -- President Dolores Marquez-Frausto.

1.02 Announcement and Public Comments regarding items to be discussed in Closed Session [Government Code Section 54957.7].

1.03 The Board will adjourn to Closed Session at approximately 5:30 p.m. Open Session will resume at the end of Closed Session in the Board Library at approximately 5:45 p.m.

**2. CLOSED SESSION**

2.01 DISCUSSION/ACTION: PUBLIC EMPLOYEE APPOINTMENT/EMPLOYMENT [Government Code Section 54957]: Title: (a) Principal.

2.02 DISCUSSION/ACTION; PUBLIC EMPLOYEE DISCIPLINE/DISMISSAL/RELEASE [Government Code Section 54957].

**3. RECONVENE TO OPEN SESSION -- DISTRICT OFFICE BOARD ROOM**

3.01 CALL TO ORDER / ROLL CALL / PLEDGE OF ALLEGIANCE BY PRESIDENT DOLORES MARQUEZ-FRAUSTO.

3.02 REPORT OF ACTION TAKEN IN CLOSED SESSION.

**4. PUBLIC HEARING**

4.01 PUBLIC HEARING: Public Hearing at approximately 5:45 p.m. to solicit the recommendations and comments of members of the public regarding the specific actions and expenditures proposed to be included in the District's Local Control and Accountability Plan or annual update to the local control and accountability plan. A copy of the District's Proposed



# References

- Altman, N. S. (1992). “An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression.” *The American Statistician* 46(3): 175–185.
- Breiman, L. (2001). “Random Forests,” *Machine Learning*, 45(1), 5-32.
- Grimmer, J. (2016). “Applying Methods to Documents.” *Machine Learning for Social Scientists*. Stanford University. Lecture.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An Introduction to Statistical Learning*. New York: Springer.
- Jurafsky, D. (2016). “Text Classification.” *CS 124 — From Language to Information*. Stanford University. Lecture.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. and Duchesnay, E. (2011). “Scikit-learn: Machine Learning in Python.” *Journal of Machine Learning Research*, 12: 2825-2830.