Introduction – 1000 / 3000 (with prep) words

- Very brief paragraph outlining project (essentially an abstract) before 1.1
- Initial proposal
    - Maintain that it is the same project with a different dataset
    - Influenced the actual project carried out
    - Licenses issues
        - Quote emails
        - Initially knew University had a license, so assumed it would cover my needs
    - Motivations still largely same
        - Unique
            - Long texts
            - British sentiment analysis
            - Sentiment analysis of texts on war
            - Finds words/phrases that are pro/anti-war
        - Political
        - Sentiment analysis (NLP)
        - Real world
        - Iraq
    - Set me back time
    - Had to work quicker
    - Had to come up with new project timeline
    - Not same problems with Hansard – Open Parliament License
        - Look into why it's not personal data of MPs
        - Quote License directly
- Motivation
    - Interest
        - Had read about NLP but never implemented anything
            - Therefore no experience with the libraries I'd be using
            - Link to later in preparation?
    - Same as before
    - More unique/important
        - Idea came from US rep-dem paper
        - Lack of accountability of politicians
        - Data is publicly available, but in a bad form, so making it more usable is good
        - NLP on Hansard
            - Great data, presented badly
                - Difficult for us to understand day-to-day of house of commons where our MPs our meant to represent us.
                    - Cite some articles about these problems
                - Retrieved and cleaned up this data for the project
                    - Under open parliament license I can (and will) create an API for this data, so others can create applications to solve more of these problems
    - Relevance
        - Chilcott enquiry was recent

            - Discuss findings
        - Similar issues in Syria
            - MPs voted to bomb Syria despite the will of the public
- Challenges
    - Necessary to consider political context in order to write machine learning algorithms (in particular sentiment analysis). [quote and cite political AI article/paper].
    - There are many flaws of our 'democracy' in the United Kingdom. In my opinion, one major flaw is our voting system; we have a first-past-the-post system in which each member of the electorate only gets one vote. This vote goes to a candidate who is (usually [cite]) a member of a political party. This party will often force an MP to vote in accordance with the party line, regardless of the MP's own views. This subverts democracy when MPs vote with their party line, despite making contradictory election promises [cite? Find stats?]. This issue was particularly prevalent leading up to the Iraq war, where despite protests [cite] showing the public's overwhelming opposition to the invasion of Iraq, MPs voted in favour of the invasion. Within a month of being elected as Prime Minister in 1997, Tony Blair said "Mine is the first generation able to contemplate the possibility that we may live our entire lives without going to war or sending our children to war. That is a prize beyond value." [cite], just six years before encouraging his Labour Party to vote invade Iraq. As a result, many Labour MPs who were previously opposed to the invasion voted in its favour. This hypocrisy of Members of Parliament is commonplace in British politics, meaning that as the electorate, we are frequently misled by the politicians representing us and the difficulties of party politics add to this.
    - Manually labelling speeches is cumbersome and not possible within this project (give number of relevant speeches). Manually labelling MPs' stances is also very time consuming, as it would require a lot of research reading old newspaper articles, and even then, the stances of some of the lesser known MPs might still be ambiguous. Therefore, in this project I will have to use MPs' voting records to label their speeches. Due to reasons I have previously mentioned [reference back], MPs will not necessarily vote consistently with their own views, meaning that there will be some degree of noise in the data. This issue is particularly great for the Iraq war, since the debate split parties and the leadership of both the Labour Party and Conservative Party were in favour of the war; before the Iraq war vote on 18th March 2003, the leaders of the two largest parties, Tony Blair (Labour Party) and Iain Duncan-Smith (Conservative Party) used party whips to persuade MPs to vote to support the invasion.
    - One result of this project will be to provide evidence in support of one of the following hypotheses:
        - Hypothesis 1: MPs who feel strongly enough about a given issue to speak about it in the House of Commons will not vote on their own convictions, rather than in accordance with their Party's line.
        - Hypothesis 2: Political Parties can influence their own MPs to vote with the party line, regardless of the views of the individual MPs.
    - Admittedly, viewing the issues in such a binary way is an oversimplification and how an MP's party affects their vote will differently vary on a vote-by-vote basis for different MPs. While this simplification would be naïve in a Politics dissertation, overlooking nuances will allow for greater technical discussion, relevant to

    Computer Science dissertation. In essence, the better the performance of the classifier produced, the greater weight will be given to Hypothesis 1 over Hypothesis 2.
  - The noise in the data is taken into account by adjusting the hyperparameters
- Previous work – look these up
  - Break this section down by type of other work
  - What are the problems with these works?
    - Lack of British politics
    - No NLP on Hansard
    - Long text sentiment analysis
    - But they suggest that if the data is good, it should be a good project
  - Hansard studies
  - Rep-dem paper
  - Twitter sentiment analysis
    - Shorter texts
  - Sentiment analysis in the news
  - Iraq war media study (initial dataset). Mention results
  - UK parliament research/articles about following party lines
  - Look for other long text sentiment analysis
    - Show SVM > neural network
  - SVM BoW model?
  - Entailment for politics?

Preparation – 1200 / 3000 (with intro) words

- Goals
    - Some same as original proposal
    - Scape transcripts
    - Clean-up transcripts
    - Collate transcript and voting data
    - Create database
    - Create classifier
        - Better than Naïve Bayes
    - Evidence for one of the Hypotheses given in the introduction
    - Learning about correlations and differences in speech between pro-war and anti-war language
    - My learning
- Expected outcomes
    - Somewhere between the two hypotheses, as some MPs will be more influenced by their parties than others
        - More likely to rebel (against the party line) if they have no intentions of rising up the ranks of the Party
    - What tense should I write in?
- Changes from original proposal section?
- Make it clear implementation wasn't trial and error
    - Development plan
        - Link to requirements analysis
        - Specific success criteria associated with each component/milestone
        - Agile
        - Consider testing and evaluation throughout
            - Easy because of F1
                - Cross validation
                    - Means don't have to use train data
                    - Not overfitting
        - Planning testing
            - No problem with human subjects for MPs. Why? Secondary data?
            - Unit testing
            - Spam email
            - Train/test split
            - How do human factors affect testing?
                - Idea is to mitigate them, so this isn't a problem
            - Have confidence metrics in evaluation
            - What is the final metric to consider?
                - F1 of _____
- Things I had to learn:
    - Define all relevant terms
    - Provide context relevant to this specific project throughout these sections
    - Relevant courses
        - Use course books as references
        - Machine Learning and Bayesian Inference
        - Natural Language Processing

            - Information Retrieval
            - Databases
            - Artificial Intelligence
            - NST maths
            - Had to learn the Part II courses in advance of the lectures
        - Python more
            - Had only done large Python projects using the Django framework
        - NLP techniques
            - Sentiment analysis
            - Stemming
            - Entailment
        - Machine Learning
            - SVM
                - Diagrams
                    - Kernel functions
                    - Hyperparameter importance
                    - Support Vector diagram (ie 2D graphical representation)
            - SVD/PCA
                - Diagram
            - F1
                - Precision & recall
- What project proposal would have looked like, had I started with this dataset
    - Timeline
        - Plan everything for presentation and professional practice
        - Extra work required for data retrieval
        - Had to work faster
    - Description of starting point with this project
        - Include code written by others
            - Hansard compiled
            - Libraries
                - Justify why to use them over own code
            - Voting data
    - Success criteria
        - Should be clear
    - Changes to the extensions
- Research into parliament
    - Manually using the Hansard
        - Screenshot showing problems with scraping from it
    - Speaking to Chris Smith
        - Gave me context and intuition as to how language differs
- Introduction to Bag of Words model
    - Mathematical reasoning for why BoW works?
    - Include stuff about stop word removal, stemming and n-grams
    - Cite NLP book
- Introduction to Entailment
    - Short section, highlighting that it’s a different way of viewing the same problem
- Introduction to Naïve Bayes

        - Baseline
- Introduction to Support Vector Machines
    - With first bit about classifiers in general
- Introduction to Cross-Validation (or include in SVM section?)
    - Diagram
    - Include stuff about F1?
    - Briefly describe using a validation set alternative and explain why not using this over cross-validation
- Database Schema
    - Developed by looking at the datasets and thinking about what data would be useful/necessary for the task and which queries would be most common to optimise those
    - What mathematical properties does the schema have?
        - Normalised? Look-up what this means
    - Provide Entity-Relationship diagrams
- Requirements analysis
    - Written given background knowledge
        - Link to introduction and the rest of the preparation section
    - Break components of project down
    - Include time estimate, priority and difficulty of each part
    - Use to justify plan
    - What needed to be worked on and how was it prioritised?
        - Data scraper first
        - Then Database
            - Not 100% necessary, but speeds up processing, as it means that data doesn't have to be scraped over and over every time it is needed
        - Then classifier
        - Then extensions/optimisations
    - Dependency graph
        - Shows how project depends on other data/modules
            - Include a bit about which elements in the dependency graph are replaceable
                - E.g. can replace some libraries with other alternatives
                - Can't replace the data
- Training/test split
    - Proper practice
    - Didn't look at/use test data
- Spam Email Dataset
    - Justify why this specific dataset
        - Good established results
            - Similar method?
        - Explain how it's essentially the same task
    - Explain how it's essentially the same problem
        - Also mention differences and how these affected implementation
    - Helped me learn
    - Good practice – not hard coding anything specific to the task

- Good way to evaluate the model
    - Difficult to evaluate since first sentiment analysis on this data
    - But there are established results for spam email dataset that I can compare to my results on the same dataset (like-for-like comparison)
        - Mention how results in the spam paper will be better due to the fact that they are tailored to that dataset
            - E.g. adaptive model. This wouldn't make sense for the context of the Iraq war data, so not worth doing. But I planned to do other (context-specific?) optimisations (e.g. entailment)

- Choice of tools
    - Give table of all tools, versions and licenses used
    - Abstract discussion of what the tool needs to do, then concrete explanation of how the given tool fits this (and its limitations where they exist)
    - Link things to development model (ie probably agile)
    - Retrospective discussion of where things proved useful in this specific project
    - Language:
        - Python
            - Why best?
            - NLP libraries
            - Lots of community support
            - Have used before
            - Suits agile development – quick to get something together quickly
            - 3.6 because latest stable
        - Other alternatives? Briefly discuss
    - Libraries
        - Justify using them over writing own code
            - Faster
            - More reliable
            - Existing support
            - Tested
            - But less task-specific/versatile
    - Database
        - SQLite
            - Why best?
            - Simple to use
            - Works well with Python
            - Lightweight
            - Lots of support
                - DB Browser for SQLite
            - Knew SQL already
            - Its limitations didn't matter
                - Security [cite]
        - Other possibilities? Look-up
            - Django?
            - NOSQL?
    - Backup

        - GitHub
            - Used before
            - Links to git
    - Revision Control
        - Git
            - Used before
            - Lots of support
        - Stores code and LaTeX
    - Development environment
        - Visual Studio Code
            - Used before
            - Lots of extensions (versatile and modifiable), due to the fact that it's open source
                - Git
                - PyLint
        - PyCharm
            - Why not? License wouldn't cover?
    - Hardware
        - My own computer for development & testing
        - MCS computers as backup
- Software engineering Techniques
    - Iterative? Agile? Extreme? Look at old lecture notes for this
        - Are there other ways that work for 1 person?
        - Agile makes sense, as not sure whether the development of a classifier using the data will be possible, so this gives results as quickly as possible
    - PyLint for coding standards
        - Give examples of what PyLint does
    - Documentation
    - Print statements (/program trace)
    - Unit tests
    - Test/train split
    - Spam email dataset
    - Modular
        - Use of settings to control flow
            - Cross-validation determines these settings, so they are not hard-coded
        - Easy to add extensions
    - Summary
        - Table of sprints and estimated durations

Implementation – 800 / 4500 words

- Section for each component?
- First section is an overview?
- Throughout, talk about:
    - Uniqueness
    - Challenges
    - Context
- High-level program flowchart/state diagram
    - Other more specific flowcharts?
- Decisions on which data structures were used
    - Where applicable, write about low-level hardware implementation of data structures
        - Diagrams
    - Talk about complexities of operations
    - Briefly mention possible alternatives and justify choice
    - SQLite3
    - Set()
    - collections.Counter()
    - numpy
        - Speed
- Include code
    - Comment a lot
    - Interesting code. Don't include code snippets for all of them, but describe all?
        - N-gram
            - Get_n_grams()?
        - Get_mp_folds
            - Code for get_member_no_of_speeches()
            - For stratified CV
                - Quickest way to maintain randomness
                    - Shuffle and loop until it fits constraints
                - Include maths for total speeches
                    - Std = sigma / root(n)
                - Within 15% of average for aye percentage
        - Remove_stop_words()
            - Reasoning for using set() over other data structures
                - Underlying implementation
                - O() of operations
        - Something using regex (re)
            - Remove tags in .ems
            - Remove punctuation
                - Maybe not full function if I use this, since the entire function could probably be a more complex regular expression
            - Remove tags in Hansard
        - Generate_word_list()
            - Shows good use of settings
        - Condense_bags() vs svd code
            - Common words generated for train

      - Common words passed in for test
      - Timeit?
    - Database insertions
      - Error handling
    - Remove titles
      - Some have multiple titles
      - Give examples
    - Match_full_name()
      - Match_list for MPs that couldn’t be matched
      - Also sped up matching for other MPs (match only had to be done once)
    - Match_first_and_family_name()
      - Change to give argument to MatchException
- What was produced
  - Data scraper
  - Database
  - Bag of Words classifier
  - Entailment system
  - Each of these is a major milestone and a section of a sprint
- Design strategy looking to testing
  - Same things mentioned in other places. Reiterate.
    - Unit testing
    - Constantly using F1 as metric throughout
    - Not ever testing on test split until the end
    - Naïve Bayes
    - Spam dataset
- Libraries used
  - Discuss which used and why (including why not using own code)
    - NLTK
      - N-gram
      - Porter stemmer
        - Discuss possible alternatives
          - Give examples of how different stemmers stem words
      - Stop words
    - BeautifulSoup
      - Can use for spam dataset and Hansard scraping
        - Versatile
    - SKLearn
      - Gives probability in SVM, which can be used to find probability of a speech being pro/anti-war. These probabilities can then be propagated and used to determine an MP’s stance
      - Easy
      - Modifiable
        - Including cache size for speed-ups
      - Good documentation
      - Good community support

            - Why not Tensor Flow?
            - Used:
                - Svm (.SVC())
                - F1_score
                - Accuracy_score
        - Random
            - Shuffle
        - re
        - Pickle
            - To avoid rerunning when programs take a while to process
                - Works well as can (e.g.) save settings
            - Use for test suite
        - Numpy
            - Fast
            - Used:
                - Array
                    - Underlying hardware implementation
                        - Contiguous?
                - array_split
                - norm
                - log10
                - logspace
                - mean
                - sqrt
                - std
                - svd
                    - Much faster than SKLearn equivalent
        - SQLite3
            - Lightweight
            - Security issues don't matter
            - Fits needs
- Retrieval
    - Hansard parsing
        - BeautifulSoup
    - Hansard cleaning up
        - Json poor attempt at API
            - Was it ever called an API? Look into this
            - Non-standard sections and other problems (can't traverse between things) mean that code has to be written with lots of error handling
    - Voting dataset
    - Matching up datasets challenges
        - Recursively removing titles
        - Two match functions & blacklist
        - Levenshtein distance
            - Give maths, reasoning and example diagram
        - Some unavoidable anomalies – lords, ladies, maiden names, middle names etc. and no possible blanket rule

            - Had to do these manually
- Bag of Words
    - Spam
        - Parsing .ems files
            - Like xml files so modified (added surrounding email tags), then used BeautifulSoup
            - Other issues with BeautifulSoup
                - Changed &NAME to #NAME (and same for num, website etc.)
                - Why did this happen?
                - Call these steps preprocessing
            - Settings
            - Classifiers
            - Baseline
                - Naïve baseline
    - Commons
        - Per speech vs. per MP
            - Which is most reliable
            - Most useful?
            - Output both?
            - Use probability for per MP
        - Settings
            - Cross-validation to learn
            - T-tests
        - Classifiers
            - Processing data for classifier
            - Hyperparameter learning
                - Why initial settings learning heuristic didn't work
                - Grid search with expansion?
                    - What will I do here?
                    - Provide diagram
                - Noise of data affects hyperparameters
                    - Data is noisy due to its nature
                        - MPs don't necessarily vote consistently with their own views
        - Baseline
            - Naïve Bayes
        - Extensions
            - Intuitions about what good for each dataset
                - E.g. stop word removal good for commons, not for spam
                - Base this on actual run settings
            - Entailment
                - Question bags normalised on separate scale
        - Code efficiency
            - Limited database accesses
                - Timeit?
            - Numpy

Evaluation – 300 / 2300 (with conclusion) words

- Evidence of thorough and systematic evaluation
- Lots of graphs/tables
- Discussion of why I am using each evaluation method
- Screenshot of terminal outputs
    - Overall program
    - Test suite
- High level test code snippets
- Training/test split
    - Cross-validation
- Other good practices for machine learning
- Measures of success
    - Were original goals achieved?
        - Reference requirements analysis and original goals
        - Component by component analysis of what was achieved
            - Write about goal and success of each bit
        - Discuss each of the goals and how they were met
    - Percent accuracy
    - F1 Score
        - Of what precisely?
            - Per speech or MP? Both?
        - Compare MP F1 with and without probabilities
        - > 0 is good, since it was initially unclear as to whether this would be possible
        - The greater the F1 score, the greater the weighting to Hypothesis 1
            - Discuss what the actual result means
    - Area under curve
        - What is this? Look up.
- Baseline
    - Naïve Bayes
- Compare spam results
    - Same data so can compare
- Comparison with rep-dem results?
    - And other similar studies
- Manually checking data in database
- How to do automatic checking of database?
- Unit tests
    - Black box or white box?
- Stress/load tests
    - What would these be?
        - How do I push system?
            - No max bag size?
                - Large n-gram run time
                    - Graph of this?
- Look into tests/limits on main libraries used
- Manual checking by backing up results with articles/other research
- Tables of performances

        - Compare optimisations
- Graphs (with confidence intervals)
    - Look at other NLP/ML papers to see what graphs are good
- Most pro-war vs. most anti-war n-grams
- Some analyses of speed
- Show limitations
    - How close it was to working in a really ambitious case?
        - Speed?

Conclusions – 300 / 2300 (with evaluation) words

- Results
  - Data scraped and cleaned
  - Database compiled
  - Compare classifier to baseline
    - Shown F1 > 0
  - Context of results due to uniqueness
- Highlight uniqueness
- Discuss these results briefly
  - Link back to introduction
  - Link to academic papers/articles
  - Which initial goals/requirements were met and which were not?
    - Shown Hansard is a great dataset for NLP and have produced something that will facilitate its use further in the future
- Lessons learned
  - What would be done differently if it were done again
    - Think about licenses properly to avoid those problems
      - But I have learned through this mistake
  - Data produced
    - Easier to digest
    - What else can be done with it?
      - API
        - Licensing behind this. What does the Open Parliament License say? What do libraries I'm using say?
          - My intellectual property
  - Specifics about what it was in my model that worked well in the context and generalise to give some conclusion
  - What would be done differently next time?
  - What I learned
    - NLP
    - ML
    - Tools
    - Politics
- Future work
  - Say I will actually do a lot of this
  - API
  - Other political issues
  - Multi-class classification (with e.g. not present or abstain)
    - This is difficult to gage from speeches, as reasons someone isn't there or abstains vary a lot therefore beyond this scope
  - NN?
    - Mention why it wasn't used here
    - Manually labelling data would likely give a better improvement for this specific task
  - Web platform
    - Summarising debates and MPs' stances, votes and speeches
      - Greater accountability

        - Link API to web platform
    - Cross-correlate/use data from twitter or press
        - Look into how opinions of public, media and politicians vary over time?