**Flinders University**

**School of Computer Science, Engineering and Mathematics**

# A User-Oriented Statistical Model for Word Prediction

*Improving Hands-Free Assistive Technologies for Typing*

**By**

**Madhuvanthi Muralidharan**

A thesis submitted for the degree of Bachelor of Engineering (Biomedical)(Honours) and Master of Engineering (Biomedical)

**Project Supervisors**
Professor David Powers

October 2016

Submitted to the School of Computer Science, Engineering and Mathematics in the Faculty of Science and Engineering in partial fulfilment of the requirements for the degree of Bachelor of Engineering (Biomedical) (Honours) and Master of Engineering (Biomedical) at Flinders University – Adelaide, Australia.

# Declaration of Originality

I certify that this work does not incorporate without acknowledgement of any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by any person except where due reference is made in the text.

Signed: Madhuvanthi Muralidharan Date: 17th October 2016

# Abstract

Word prediction has seen a vast range of applications over the years including SMS smart-messaging systems, and recently, in Assistive Technologies. The primary intention of word prediction is to reduce typing time, and while most prediction systems in Assistive Technologies achieve this goal, there is a requirement for the user to type using a keyboard. This is difficult for many disabled persons, but borders impossible for users suffering with severe speech or motor impairments. To enable such persons to type, a user interface is proposed in which the user is able to type by using predictions – at least from the users' perspective – eliminating the requirement of a keyboard. This interface is part of a broader project on Unconscious Computer Interface where Brain Computer Interfaces are used to detect implicit intention rather than requiring explicit action.

The aim of this project was to build a word-prediction model for the aforementioned user interface. Most word prediction models are based on statistical language models, in particular the *n*-gram model. Many improvements have been suggested to improve upon its performance, but little focus has been placed on user-based language models. Thus, the project aimed to address this research gap by building a user-oriented word-prediction model that is capable of generating predictions based largely on the user than the language.

The model was designed as a back-off language adaptation model, drawing upon dynamic and static corpora to generate the predictions. Baseline studies validated the model as a word prediction tool, as well as identifying design elements with potential to improve performance. It is anticipated that the model will be integrated with the proposed Unconscious Computer Interface in future.

# Acknowledgements

I wish to extend my deepest gratitude to my supervisor, Professor David Powers for his continual guidance, support and kindness throughout the project. His patience when imparting his deep knowledge, not only in his field but across various disciplines, has been invaluable.

Thank you to my family, who have accompanied me on my journey so far with their unconditional love, support and enthusiasm.

# Table of Contents

# Table of Figures

# Table of Tables

# List of Acronyms and Abbreviations

| Acronym | Full Form |
|---|---|
| AAC | Augmentative and Alternative Communication |
| AT | Assistive Technology |
| BCI | Brain Computer Interface |
| BNC | British National Corpus |
| CALO | Cognitive Assistant that Learns and Organises |
| CFD | Conditional Frequency Distribution |
| EEG | Electroencephalogram |
| EM | Expectation-Maximisation |
| EMG | Electromyography |
| EOG | Electrooculography |
| ERP | Event-Related Potential |
| HMM | Hidden Markov Model |
| IR | Information Retrieval |
| KSR | Keystrokes Savings Rate |
| KT | Keystrokes Typed |
| LDA | Latent Dirichlet Allocation |
| LSA | Latent Semantic Analysis |
| MLE | Maximum Likelihood Estimate |
| NLP | Natural Language Processing |
| NLTK | Natural Language Toolkit |
| OOV | Out-Of-Vocabulary |
| PCFG | Probabilistic Context-Free Grammar |
| RSVP | Rapid Serial Visual Representation |
| SCI | Spinal Cord Injury |
| SLM | Statistical Language Model |
| SVD | Singular Value Decomposition |

| | |
|---|---|
| TF-IDF | Term Frequency-Inverse Document |
| UBC | User-Based Corpus |
| UCI | Unconscious Computer Interface |
| WER | Word Error Rate |
| WPM | Words Per Minute |
| WSD | Word Sense Disambiguation |

# List of Nomenclature

| Variable | Definition |
|---|---|
| $\sigma$ | Sample Standard Deviation |
| $\mu$ | Sample Mean |
| $x_i$ | Sample Values |
| $\lambda$ | Arbitrary Value |
| C(w) | Number of times word *w* occurs in a sample |
| C* | Laplace Modified Count |
| $d_c$ | Laplace Discount Coefficient |
| $d_r$ | Good Turing Discount Coefficient |
| N | Total Observations in Sample |
| $n_r$ | Number of events occurring *r* times in training |
| P(w) | Probability of word *w* in sample |
| P* | Discounted Probability |
| PP | Perplexity |
| R | Event occurring *r* times |
| r* | Absolute Discount Coefficient |
| $SE_{\bar{x}}$ | Standard Error |
| T | Threshold for Frequency Counts |
| V | Vocabulary Size |
| $w$ | Word |
| $w_n \mid w_{n-1}$ | Bigram |
| $w_n \mid w_{n-2} w_{n-1}$ | Trigram |

# 1. Introduction

Speech impairments and a lack of motor control seen in severe disabilities make communication a difficult if not impossible task. To enable such persons to communicate, for example through letters or documents, significant focus has been placed on the development of Brain Computer Interfaces (BCI). BCIs are an emerging Assistive Technology (AT) that are primarily used for quadriplegics and persons with related diseases in which there is either partial or complete paralysis of, part of, or of the whole body. BCIs utilize the user's brain signals as the basis for communication and selection (Schalk et al., 2004). The most popular BCI for document-production is the P300 Speller which uses an alphanumeric grid. The rows and columns of the grid are successively highlighted and once the row or column contains the target letter, a P300 signal is elicited in the user's brain. This signal is then processed and used to identify the target letter that the user wished to type (Farwell and Donchin, 1988).

The difficulty with the P300-Speller and other similar word-typing BCIs is the time taken to produce words (almost a few minutes) and their reliance on character-by-character typing (Farwell and Donchin, 1988). This often distracts the user from the overall content they wish to produce, and therefore such devices rarely see the completion of complete and formatted documents or other office-related tasks.

A way to reduce the typing time and move from the character-level to a word-level is word prediction. Predictive communication is not uncommon and has found its uses in a vast range of fields, from speech and handwriting recognition to SMS-smart messaging systems and recently, Augmentative and Alternative Communication (AAC), a form of communication AT for disabled persons. In most of these applications, the predictions are displayed as a list from which the user can select a prediction, or manually type the word.

The final interface as part of the larger project (henceforth known as the Unconscious Computer Interface or, UCI), aims to provide an interface which in itself, is an integration of word-prediction in a BCI-like AAC device. Specifically, the interface will contain a list of continuously-generated stream of predictions and allow a user to type purely by using these predictions. The user uses their eyes to navigate through the predictions, with relevant predictions based on where the eye is looking becoming larger in size and the others becoming smaller and smaller until they are greeked out – i.e., the text is reduced to a simple grey line or a thick patterned line in the case of multiple texts. It is anticipated with this interface that the requirement of any keyboard is removed and the typing time significantly reduced.

Word prediction is commonly achieved by the use of statistical language models (SLM), which determine the likelihood of a sequence of words and assign an associated probability to it. The most extensively used language model for word prediction is based on the *n*-gram model which estimates the probability of a word given its prior history, typically drawing up the last *n*-1 words to make this approximation (Manning and Schütze, 1999). The three common *n*-gram models used are the unigram (unconditioned representing a marginal or prior distribution), the bigram (conditioned on the last term) and the trigram (conditioned on the last two terms). The conditional probabilities of the word sequences are usually estimated with respect to their relative frequencies in the text, along with additional smoothing techniques to ensure an accurate probability distribution (Manning and Schütze, 1999).

Whether writing emails, reports, or simply surfing the web, we are heavily influenced by our writing style, our preferred vocabulary and the task at hand (email, report-writing). It is evident therefore, that building a word prediction model that is tailored around the user rather than simply the language can provide lengthier, more accurate and relevant word predictions.

Hence, the aim of this project is to build a user-based word prediction model, capable of predicting words based on a user's interests, style of writing and the current task being undertaken. It is anticipated that the developed word-prediction model will be used in combination with the UCI in the future, and emerge as an innovative and useful AAC device.

The thesis first establishes a comprehensive background of both relevant language models and AT for quadriplegics and locked-in syndrome patients, and a literature review outlining current methods for improving these language models and their applications into existing AT. It then moves on to the implementation of the designed model, the experiments conducted to validate its feasibility, the results obtained, a discussion of these results and lastly, conclusions derived and future work involved.

# 2. Background

## 2.1 Language Model

Predictive communication is commonly achieved by the use of statistical language models. Language models were initially developed for speech recognition applications, but have since then been used for other natural language processing (NLP) tasks including optical character recognition and spelling correction (Manning and Schütze, 1999).

The basis of a language model is to determine the likelihood or joint probability of a sequence of words $P(w_1, w_2 \dots w_n)$, and assign an associated probability to it. There are many ways in which the probability of the sequence of words can be calculated, including decomposing the joint probability into a product of conditional probabilities (chain rule),

$$P(w_1, w_2 \dots w_n) = P(w_1)P(w_2|w_1)P(w_3|w_2w_1) \dots P(w_n|w_1^{n-1}) \tag{2.1}$$

The chain rule of conditional probabilities conditions each successive word against the $n$ preceding words, referred to as the *word history*. However, this process is computationally inefficient given the excessive number of parameters required. A well-defined method of reducing the number of parameters is by reducing the word history on which the word sequence is conditioned on. This concept is reflected in the *Markov Assumption* which states that the transition from one state to the next in a first-order finite state Markov Chain is dependent only on the previous state and not on any other prior states. Thus, under the Markov Assumption, the word history simply reduces to the last $n$ -1 words (Manning and Schütze, 1999).

$$P(w_n|w_1^{n-1}) \approx P(w_n|w_{n-N+1}^{n-1}) \tag{2.2}$$

This model is called the *n*-gram model. The most common *n*-gram models used are the unigram (unconditioned representing a marginal or prior distribution), bigram (conditioned on the last term) and trigram (conditioned on the last two terms).

### *2.1.1 Estimation*

Different estimation techniques can be used to estimate the probability of the word sequences, with the most straightforward and crudest form of estimation being the maximum likelihood estimate (MLE, Manning and Schütze, 1999). The MLE for an event ∈ is the ratio between the number of times the event occurs in a training data and all the samples in the training data. That is, the relative frequency of the event, or in this case –word, in the training data.

Example

If we have a unigram model, i.e., $P(w_1)$, the MLE of $w_1$ would be (Manning and Schütze, 1999)

$$P(w_1) = \frac{C(w_1)}{N} \tag{2.3}$$

Similarly, for bigram and trigram models, the maximum likelihood estimate of $(w_n w_{n-1})$ and $(w_n w_{n-1} w_{n-2})$ would be

$$P(w_n|w_{n-1}) = \frac{C(w_n w_{n-1})}{C(w_{n-1})} \tag{2.4}$$

$$P(w_n|w_{n-1}w_{n-2}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})} \tag{2.5}$$

The shortcoming of this approach is that there is severe bias towards observed events and severe bias towards unseen events (zero probability assigned). That is, words that are found commonly in texts such as 'the' are assigned a large probability as they are seen often, where as more unique words that are unlikely to be seen in normal text are assigned a zero probability, i.e., there is absolutely no chance that this word will appear in any text. This is of course inaccurate, and to counter this bias and ensure there are no zero probabilities, *smoothing* was introduced. The general approach to smoothing is to discount some probability from observed events and redistribute it to unseen events (Manning and Schütze, 1999).

## *2.1.2 Smoothing*

### *2.1.2.1 Good Turing Discounting*

Good-Turing discounting is based on the Good Turing formula (Good, 1953), where the modified count for an event occurring $r$ times is discounted using the following discount coefficient

*Equation 1 Good-Turing Discount Coefficient*

$$d_r = (r+1)\frac{n_{r+1}}{r.n_r} \tag{2.6}$$

where $n_r$ corresponds the number of events that occurred $r$ times in training. This method requires that $n_r > 0$ and therefore should only be used for events that occurred $r$ or more times ($r$ is arbitrary).

### *2.1.2.2 Laplace Smoothing*

Laplace Smoothing simply adds one to each observed count. The modified count is therefore

(2.7)

$$c^* = (C(w_1) + 1)\frac{N}{N+V}$$

where $\frac{N}{N+V}$ is the normalisation factor (V is the vocabulary size and N is the number of tokens). The discounting coefficient is (Jurafsky and Martin, 2000).

$$d_c = \frac{c^*}{c} \tag{2.8}$$

#### *2.1.2.3 Absolute Discounting*

In absolute discounting, the event count is discounted by some constant value $\lambda$ $(0 < \lambda < 1)$. In this case, it has least effect on the higher-counts than the lower-counts. The discount coefficient is therefore defined as (Ney et al., 1994)

$$r^* = \frac{r - \lambda}{r} \tag{2.9}$$

The constant value $\lambda$ can be chosen using using held-out estimation

#### *2.1.2.4 Kneser-Ney Smoothing*

Kneser-Ney Discounting extends the absolute discounting method by providing a more refined probability distribution for the lower order n-grams by considering the number of contexts in which the word appears in. That is, the number of times a word $w_i$ completes a bigram (expressed as $\{\{w_{i-1}: c(w_{i-1}w_i) > 0\}|$). Normalizing this count with the total number of bigrams therefore gives the following probability for word $w_i$ (Martin and Jurafsky, 2000).

$$P(w_i) = \frac{|\{w_{i-1}: c(w_{i-1}w_i) > 0\}|}{\left|\{(w_{j-1}, w_j) : c(w_{j-1}w_j) > 0\}\right|} \tag{2.10}$$

#### *2.1.2.5 Model Combination*

Discounting methods are all a form of smoothing in which probabilities are artificially assigned to unseen events to eliminate zero probabilities. Another smoothing technique that is used is to combine models. For example, if there is no evidence of a particular trigram in the training text when estimating the probability $P(w_n|w_{n-2}w_{n-1})$, then its probability can be estimated by a bigram instead i.e., $P(w_n|w_{n-1})$ or even a unigram $P(w_n)$. Alternately, trigram, bigram and unigram probabilities can all be combined to allow the model to have a broader spectrum of the contexts available.

##### 2.1.2.5.1 Back-off Modelling

Back-off modelling is a technique in which the model 'backs-off' to lower-order $n$-grams when the required $n$-gram has 0 counts. Back-off modelling with a discounting factor was introduced by Katz (1987), in which a discounted probability $P^*$ is used for seen $n$-grams, otherwise the algorithm 'backs-off' to a lower order $n$-gram model. The back-off algorithm is therefore computed as

$$P_{BO}(w_n|w_{n-N+1}^{n-1}) = \begin{cases} P^*(w_n|w_{n-N+1}^{n-1}) & if\ C(w_{n-N+1}^{n-1}) > 0 \\ \alpha(w_{n-N+1}^{n-1})P_{BO}(w_n|w_{n-N+2}^{n-1}) & otherwise \end{cases} \quad (2.11)$$

where $\alpha$ is the back-off weight used to distribute the discounted probability mass to the lower-order $n$-grams. Katz smoothing is often used in conjunction with Good-Turing smoothing to provide estimates for the $P^*$ and $\alpha$ values (Martin and Jurafsky, 2000).

##### 2.1.2.5.2 Linear Interpolation

In linear interpolation, different order n-grams are combined together using linear interpolation. Thus, the probability estimate for $P(w_n|w_{n-2}w_{n-1})$ can be computed as

$$P(w_n|w_{n-2}w_{n-1}) = \lambda_1 P(w_n|w_{n-2}w_{n-1}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n) \quad (2.12)$$

where:

$$\sum_i \lambda_i = 1$$

Linear interpolation is not constrained to only combining n-gram models, but rather all types of language models. The general form for linear interpolation of language models is

$$P_{interpolated}(w_n|w_{n-2}w_{n-1}) = \sum_{i=1}^{n} \lambda_i \, P_i(w_n|w_{n-2}w_{n-1}) \quad (2.13)$$

The parameter coefficients $\lambda$ are typically set using a held-out corpus, choosing values that maximize the likelihood of the held-out corpus. One method of obtaining the optimal parameter coefficients is the iterative expectation-maximization (EM, Dempster et al., 1977) algorithm.

#### 2.1.2.5.3 Maximum Entropy

Maximum Entropy (Berger et al., 1996) models do not combine the models themselves, rather they combine selected features from the individual models into one model. The main advantage of this method is the ability to create an overall model that is essentially a combination of the strengths of each individual model. The main disadvantage is the computational inefficiency.

### *2.1.3 Language Model Adaptation*

Studies have shown that *n*-gram models trained on relevant domain-specific training sets and applied to similar domain-specific test sets tend to be more accurate than using generic training sets applied to a domain-specific test set. For example, in modelling casual phone conversation, it was found better to use a training set of '*text has been removed due to copyright reasons'* (Bellegarda, 2004, p. 94). Language model adaptation aims to capture this by considering two main text corpora – the smaller adaptation or *dynamic* corpus A, and the larger *static* background corpus B. The general framework for SLM adaptation is shown in Figure 2.1 (Bellegarda, 2004). From this framework, the idea is to use Corpus A to "adapt" the background corpus B based on 'task-specific knowledge' that can be derived from Corpus A that is, get a task-specific SLM from Corpus A and combine it with the static background SLM from Corpus B to output an overall adapted SLM.

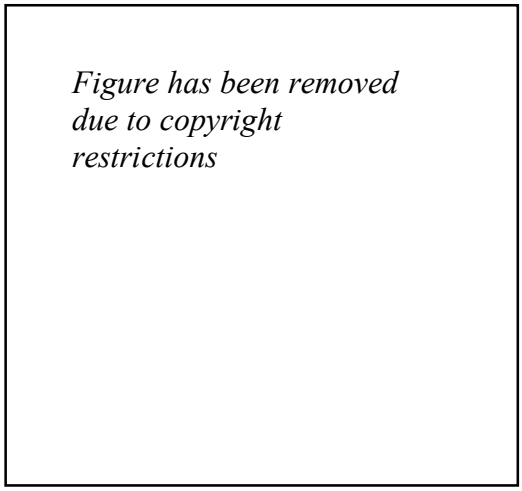


*Figure 2.1 Framework for SLM Adaptation (Bellegarda, 2004)*

There are several ways of combining these two SLMs including linear interpolation and back-off modelling. The linearly interpolated model combines the dynamic and SLM as,

$$\mathrm{P}\left(w_q \middle| h_q\right) = (1-\lambda) Pr_A\left(w_q \middle| h_q\right) + \lambda Pr_B\left(w_q \middle| h_q\right) \tag{2.14}$$

where the interpolation coefficients $\lambda$ are found using the EM algorithm and $Pr_A(w_q|h_q)$ and $Pr_B(w_q|h_q)$ are the $n$-gram estimates based on Corpus A and Corpus B respectively.

Meanwhile, the back-off SLM adaptation essentially backs-off from the dynamic SLM estimate to the static estimate depending on the frequency count. This is referred to as the fill-up technique, with the implementation

$$\mathrm{P}(w_q|h_q) = \begin{cases} P_A(w_q|h_q) & if\ C_A(h_q w_q) \geq T \\ \beta P_B(w_q|h_q) & otherwise \end{cases} \tag{2.15}$$

where $T$ is some threshold for frequency counts (for the designed model, 0) and $\beta$ is the back-off coefficient.

## *2.1.4 Language Model Evaluation*

### *2.1.4.1 Perplexity*

A common method to evaluate a language model is to measure the perplexity of the trained model on some unseen test-data (Martin and Jurafsky, 2000). Perplexity can be defined as inverse probability of a test set given by the language model, normalized by the number of words in the test data. The equation for perplexity for a bigram model is

$$PP(W) = \sqrt{\prod_{i=1}^{N} \frac{1}{P(w_i|w_{i-1})}} \tag{2.16}$$

Hence, the smaller the perplexity, the larger the probability assigned to the test data and the better the model.

### *2.1.4.2 Keystroke Savings*

Keystroke savings rate (KSR) is a commonly used method for word prediction language models. As the name suggests, keystroke savings rate is a measure of the number of keystrokes saved when typing some unseen test data (Trnka and McCoy, 2008). The larger the KSR, the more accurate the predictions and the better the model. The equation for keystroke savings is

$$KSR = \frac{Total\ Entered\ Characters - Keys\ Pressed}{Total\ Entered\ Characters} \times 100\% \tag{2.17}$$

### *2.1.5 Stopping and Stemming*

Stopping and stemming are popular information retrieval (IR) techniques (Baeza-Yates and Ribeiro-Neto, 1999). Stopping refers to the elimination of common words such as 'for', 'and' and 'a' in a query with the argument that these words do not contribute to the general meaning. This argument is also applied to words that have multiple prefixes or suffixes i.e., 'like, likely, likeable' etc. Stemming, therefore, eliminates the prefixes and suffixes of such word to retrieve the 'stem' of the word, in this case, 'like'. These techniques can be used to generate predictions based on semantic relations given that the function words have been filtered leaving only content words.

## 2.2 Application

### *2.2.1 Spinal Cord Anatomy*

This section explores existing AT devices available for patients with severe disabilities. These disabilities often stem from spinal cord injuries (SCI) or inherent neurological disorders. The first part of this section looks at introducing basic anatomy to understand the underlying causes of SCI and other relevant disorders, before moving on to existing AT.

The spinal cord is the largest nerve in the body that extends from the brain down to the waist. It is protected by sections of bone referred to as vertebrae, which together, make up the spinal column. The vertebrae are generally grouped as

cervical (upper region), thoracic (middle region), and lumbar and sacral (lower regions).

The purpose of the spinal cord is to act as a communication channel between the brain and the body (Shepherd Centre, 2011). Voluntary motion is achieved by the conduction of nerve impulses from the brain to the spinal cord, and then to the body via peripheral nerves to initiate movement. Similarly, sensory function is achieved by carrying sensory stimuli from the body, through the spinal cord and to the brain to analyse the various sensations (pain, temperature, touch etc.,) (Brodwin et al, 2009). Each segment of the spinal cord is responsible for innervating a different region of the body. Figure 2.2 illustrates the motor-sensory map of the spinal cord.

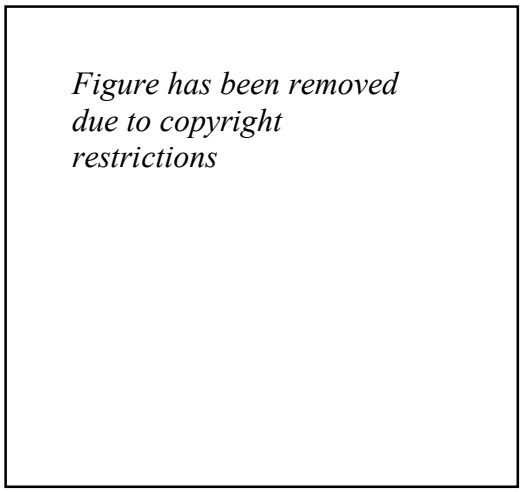


*Figure 2.2 Motor and Sensory Map of the Spinal Cord (Spinal Hub. 2016)*

### *2.2.2 Quadriplegia*

SCI occurs when there is any injury or damage to the spinal cord affecting the communication channel between the brain and the rest of the body. This affects the sensory, motor and autonomic function below the level of injury. Generally, the higher the injury on the spinal cord, the more dysfunction the individual will face (Shepherd Centre, 2011).

Quadriplegia is a type of SCI in which the cervical nerves are damaged. It is typically categorised as paralysis of the upper and lower extremities, however injury to the different cervical nerves will have a varying effect on the paralysis, and other bodily functions. Injury to the high cervical nerves (C1-C4) is the most severe, causing partial or complete loss in the arms, hands, trunk and legs, as well as respiratory, speech, bowel and bladder impairment. Damage to the lower cervical nerves is usually not as severe, with some individuals retaining some level of movement in hands and shoulders, as well as breathing and speech (Shepherd Centre, 2011). The causes of quadriplegia can be due to traumatic injuries such as car accidents, falls or sport-related, or illnesses including cancer, stroke, cerebral palsy and multiple sclerosis (Shepherd Centre, 2011).

### *2.2.3 Locked-in Syndrome*

Locked-in syndrome is a rare neurological disorder in which the individual is completely paralysed of all voluntary movement, excluding movement of their eyes. The primary cause of locked-in syndrome is damage to a specific part of the brainstem called the pons, specifically the ventral pons. Damage to the ventral pons in turn damages the corticospinal tracts causing quadriplegia (Laureys et al., 2005).

### *2.2.4 Existing Assistive Technologies*

To enable persons with severe speech or motor impairments such as quadriplegics or locked-in syndrome patients to communicate, significant research has been focused on developing appropriate AT. Examples of such technologies range from simple switches, sticks or pointers that can be used by the hand, tongue, chin etc., to gaze and motion tracking interfaces using electrooculography (EOG), electromyography (EMG) and electroencephalogram (EEG). Amongst these, the communication devices applicable to both locked-in syndrome patients and quadriplegics are the interfaces using EOG, EMG and

EEG (Guerreiro, 2007).

Broadly, all three technologies essentially use the electrical signals produced by the eye, muscle or brain respectively to control an external device. For the purpose of this project only EEG-based technologies will be considered henceforth, as they are an established AT for text-production i.e., the P300-Speller.

### *2.2.5 Brain Computer Interfaces*

The most common application of EEG-based AT is the BCI. The basic design of a BCI is to first acquire the brain signals using electrodes, then process the brain signals to extract features of interest relating to the user's intention, and finally translate the features into device commands (Schalk et al., 2004). A common feature of interest used to identify the user's intent is the P300 event-related potential (ERP). The P300 signal, characterised by its positive-inflection with a latency of 300 msec, is elicited using the 'odd-ball paradigm' in which two stimuli are presented in a random order with one appearing frequently and the other infrequently. Studies have not only shown that these infrequent stimuli elicit a P300 signal in the brain, but also found a direct relationship between the amplitude of the P300 signal and the relevancy of the events (Farwell and Donchin, 1988).

The P300-Speller is word-typing BCI which exploits the P300 signal to identify the target letter that the user wishes to type. The P300-Speller consists of an alphanumeric grid containing the alphabet, along with other numbers and characters (Figures 2.3-2.4). The rows and columns of the matrix are flashed randomly and the user focuses on the target letter they wish to type. The row and column containing the target letter are the 'relevant' events while the remaining row and columns are the 'irrelevant' events. When the row or column containing the target letter is flashed, a P300 response is elicited in the brain. The amplitude of the P300 signal is detected and analysed, and the target letter

is identified as the letter at which the intersection of the row and column yielded the largest P300 signal (Farwell and Donchin, 1988).

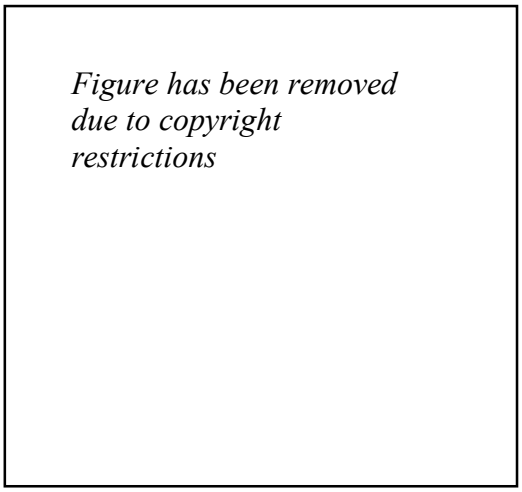


*Figure 2.3 P300-Speller, Vertical Flashes (Manyakov et al., 2011)*

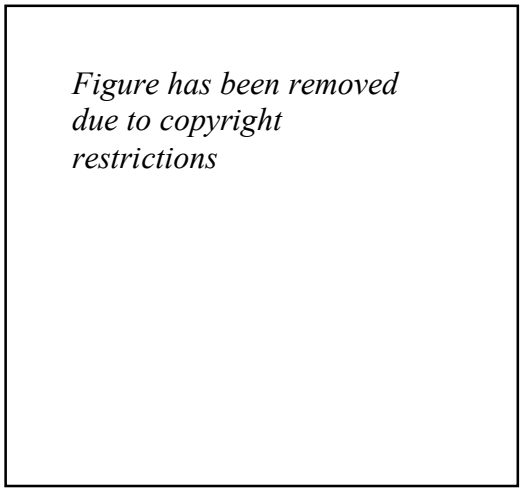


*Figure 2.4 P300-Speller, Horizontal Flashes (Manyakov et al., 2011)*

# 3 Literature Review

This literature review has two main focuses, a computational linguistic section and a biomedical application section. Both sections are based on language models, where the former section explores techniques utilized in improving a standard language model for word prediction, and the latter identifies the applications of language models in AT.

## 3.1 Computational Linguistics

### *3.1.1 Syntactic Models*

Syntactic language models incorporate syntactic information to language models and are widely used, particularly in speech recognition. Deriving and implementing syntactic models is a difficult task, reflected in its relatively smaller research field. The most common method of achieving a syntactic statistical model is to incorporate a statistical parser, which models a word sequence W and a parse tree T as a joint probability distribution $P(T,W)$ (Manning and Schütze, 1999).

Two main ways of integrating statistical parsers in SLMs is to use a generative or discriminative language model. In a generative language model, the statistical parser can be incorporated using $P(W) = \sum_T P(T,W)$. That is, the parser assigns a probability to all the words or sentences in the language given some probabilistic context-free grammar (PCFG). This approach has been widely used including Charniak (2000) when implementing a language model based on the "immediate-head" parser, Chelba and Jelinek (2000) who had a similar implementation to Charniak except the use of left-to-right parsing and Roark (2001) who used a top-down parsing algorithm interpolated with a standard trigram language model. All the methods found that they were able to improve the accuracy of recognition in speech recognition tasks across broad-domains.

On the other hand, Collins et al., (2005) took a discriminative approach where a first statistical parser is used to establish an initial ranking of a set of parse trees for each sentence with associated probabilities. The discriminative statistical model then re-ranks the initial parse trees by using additional features of the trees. This can be used for example, to choose the most likely recognition hypothesis in a speech recognition system based on features from the word sequences and parse trees. With regards to word prediction, the statistical parser described can be used in combination with a language model to identify the most likely word based on a pre-determined PCFG.

While statistical parsers still remain the most popular method of designing a syntactic model, a more recent approach was proposed by Kaufmann and Pfister (2012) to incorporate formal grammars with statistical models instead. The benefit of this approach is the ability to impose harder and larger number of linguistic-based constraints on the parser, which is not possible in a statistical parser whose only restriction is that it must make structural sense.

### *3.1.2 Semantic Models*

The most common method of extracting semantic information from a body of text is Latent Semantic Analysis (LSA, Landauer and Dumais, 1997). LSA is a word-similarity vector model that is used to measure semantic similarity between words based on their co-occurrence (words that occur together or with similar words). An LSA model is trained on a corpus of documents to create a $term \times document$ matrix. The matrix is reduced using singular value decomposition (SVD) decomposing the original matrix into a $term \times term$ co-occurrence matrix T. The semantic distance of any two words is found by calculating the dot product of the two corresponding vectors in matrix T. LSA uses a "bag-of-words" approach and therefore lacks the ability to maintain word order and therefore need to be combined with $n$-gram models such that some form of grammatical structure is used when predicting words.

A simple way of combining the two is linear interpolation, however, it was found that the predicted words often lacked syntactic coherence. Coccaro and Jurafsky (1998) opted to use geometric interpolation as an alternate means of combination. This non-linear interpolation meant that a higher probability could be given to words which were both syntactically and semantically likely and lower probability if either LSA or *n*-gram model deemed it unlikely. The final combined model reduced perplexity by 12%. A similar method was used by Bellegarda (2000).

Li and Hirst (2005) also combined a semantic model and a basic *n*-gram model using a similar technique in which the combined model had both an *n*-gram component and semantic associated component. Depending on the degree of semantic association of the predicted word with the prior history, either only the *n*-gram or both the *n*-gram and semantic model were used to make the prediction. Instead of using LSA for obtaining semantic information however, an alternate method using word-occurrences and then filtering them with a WordNet-like method was used where WordNet is a manually designed taxonomy of English words (Miller, 1995). Compared to a baseline syntactic language model developed by Fazly and Hirst (2003), the combined semantic language model was able to improve the keystroke savings rate by 14.63%. Additionally, the inclusion of out-of-vocabulary (OOV) entries in the semantic model was also found to have a profound impact on results.

Two other methods of integrating LSA with a standard language model were investigated by Wandmacher and Antoine (2008). The first method used a semantic caching language model. Semantic caching extends a cache language model with semantic information. Cache-based language models (Kuhn and Mori, 1990) increase the probability of a word based on its previous appearances in the long-term history, and are generally the result of interpolation of a cache *n*-gram model and a standard *n*-gram model. Semantic caching involves calculating the cosine similarity between each word and its nearest neighbours in a given context. If the value is above a certain threshold, the word is added to

the cache model and discarded if its below. The second method is partial re-ranking. This method selects the *n* best words from the standard model (i.e., content words) for LSA to first calculate their semantic similarity, and then assign a corresponding value in addition to its base probability. Both methods were able to reduce perplexity from a baseline 4-gram models.

Latent Dirichlet Allocation (LDA, Blei et al., 2003) is used to find topics in a particular document using a similar "bag-of-words" approach to LSA. Broadly, the algorithm works by first going through each document and randomly assigning each word in the document to a random topic for some fixed K topics. To improve upon the original random topic assignments, the algorithm goes through each word in each document to update the topic assignments based on (1) how prevalent the word is across topics i.e., P(word|topics) and (2) how prevalent the topics are in the document P (topic |document).

Though both LDA and LSA are similar in the sense that they both focus on content words rather than function words, LDA is typically used for topic-modelling while LSA is used for semantic analysis. To test the capabilities of LDA in semantic analysis compared to LSA, Mitchell and Lapata (2009) used both techniques to extract semantic information, combining the information with a trigram model using additive (linear interpolation) and multiplicative methods. Multiplicative interpolation is implemented using a semantic factor which determines the influence of the trigram model on the overall model. The resulting four models (additive and multiplicative for both LSA and LDA) were compared with a baseline trigram model. Multiplicative LSA-trigram model outperformed both additive and multiplicative LDA-trigram models suggesting the use of LDA is not suited for semantic modelling.

Though the combination of LSA with standard *n*-gram language models is the most widely used technique for generating semantic language models, it is not the only one. Erdogan et al., (2002) presented the semantic concept-based language model and semantic structured language model, both of which use

alternate means of measuring semantic similarity and integrating language models. In concept-based modelling, words with semantic similarity are grouped as a 'concept'. For some given word, the most likely concept sequence P(C) is found and an *n*-gram model is used to estimate P(W | C). In semantic structured language modelling, semantic classing is used to tag words according to their meaning thereby providing basic semantic relations. Statistical parsers build on this to derive more complex relations. The semantic information is then combined into a maximum entropy model. The maximum entropy model outperformed the concept-based model with regards to word error rate (WER), with most WER reduction achieved with the combination of the ME and concept-based models.

### *3.1.3 Topic Models*

Topic models are integral to language models and word prediction models to ensure predictions are within the topic of discussion. Lesher and Rinkus (2002) designed domain-specific *n*-gram models to evaluate if dynamically swapping these models in-and-out to match changing topic of conversation could improve prediction performance. They found that topic-specific models could indeed provide significant improvements in word prediction, highlighting that this improvement could be extended to using models capturing various other domains such as style, formality or genre.

The previous section introduced the idea of LDA for topic modelling. However, like LSA, using LDA solely for prediction is inaccurate as word order is not considered and like LSA, an intuitive solution is to simply combine it with an *n*-gram language model (Wallach, 2006). As predicted, the prediction accuracy was increased with this combined model over a general LDA model.

However, using LDA with language models can be computationally challenging and difficult to implement and therefore simpler methods of using topic models for word prediction have been suggested.

Trnka et al., (2005) used topic modelling to act as a filter for the predictions from *n*-gram models by increasing the probability of related words and decreasing the probability of unrelated words, given the current topic. Topic identification was achieved using the popular information retrieval technique – TF-IDF (Term Frequency – Inverse Document Frequency). This technique in IR is typically used to rank documents by focussing on content rather than function words. In this case, it is used to eliminate stop words thereby aiding in topic identification. It was hypothesized that this topic word prediction system would be able to decrease the number of keystrokes. However, while the predictions were more accurate and appropriate, the number of keystrokes did not decrease. Additionally, the substantial amount of memory and computational inefficiency of topic identification and modelling did not make this prediction system viable.

Kneser and Steinbiss (1993) used an adaptation model to modify language models based on the writing style. The adaptation model consisted of $K$ linearly interpolated language models trained on different genres and writing styles with associated interpolation parameters that were dynamically calculated based on the preceding text. The interpolation parameters were used to select the model that would be used to predict the next word based on the model that the previous text most likely came from. Results showed the reduced perplexity however, could be due to the overtraining of the adaptation model.

Similar to Trnka et al. (2005), Mahajan et al., (1999) also used popular information retrieval techniques to incorporate topical information to *n*-gram language models. The first stage was to collect all similar documents in a training database based on the current document using the aforementioned TF-IDF technique. The similar documents were then used in combination with a topic-independent trigram language model using linear interpolation. The proposed model reduced the perplexity of the baseline trigram language model by 37% with suggestions presented of using dynamic topic models with varying amounts of documents to increase robustness while maintaining specificity.

Other means of capturing contextual information are cache-based models and trigger models (Rosenfeld, 1996.). Cache-based language models were described in the previous section. A trigger language model uses word trigger pairs in which the trigger word increases the probability of its associated target word. Selecting useful trigger pairs was achieved using the average mutual information technique. Trigger language models combine word triggers and *n*-gram constraints using the maximum entropy model.

### *3.1.4 Combination of Models*

Section 1 illustrated the performance gains when either syntactic, semantic or topic information is incorporated into language models for word prediction. However, intuitively, the greatest performance improvement will occur when using a model that is able to enforce syntactic, semantic *and* topical constraints on their word predictions. Several techniques for combining language models have been proposed over the years. As described in the background section, the most commonly used combination techniques include: linear interpolation, backing-off, language adaptation and maximum entropy (Rosenfield 1994, Berger et al., 1996), most of which have been used to combine semantic models with a standard language model.

The four main combinations of syntactic, semantic and topical information are syntactic-semantic language models, syntactic-topic models, semantic-topic models, and lastly, syntactic-semantic-topic language models. Within this, combinations that have been used directly for enhancing prediction language models are syntactic-topic models and syntactic-semantic models. While it is a relatively unexplored field, few notable contributions exist.

Boyd-Graber and Blei (2009) developed a syntactic topic model that is capable of predicting words that make sense both topically and syntactically. The proposed method assigned a distribution to each word in a sentence that is a combination of topic weights (extent to which document is about a certain topic) and syntactic transitions derived from a parse tree. The model was able to reduce perplexity

compared to a hierarchical Dirichlet process (an extension of LDA which does not require pre-defined topics) and produce words that were syntactic and thematically accurate.

Griffiths et al., (2005) designed a generative model that used short-range and long-range dependencies to identify semantic and syntactic classes. Essentially, a Hidden Markov Model (HMM) is used to identify the short-term dependencies or function words and a topic model to handle long-range dependencies or content words. The results validated the models' capability of extracting functional and content words (therefore identifying syntactic classes and semantic topics). While this model was not specifically targeted at producing word predictions, the extraction of function and content words is an important precursor and can be extended to prediction systems.

Meanwhile, Khudanpur and Wu (2000) proposed a SLM which is a combination of semantic relations found by inference of topics using information retrieval techniques, and syntactic structure using a left-to-right parser. A maximum entropy model is then used to combine the two together as a single model. Results showed a perplexity and WER reduction compared to a baseline trigram model.

## 3.2 Biomedical Application

Word prediction models are used in a range of applications including AAC devices. Most high-tech AAC systems use some form of word prediction, for example Word Q (Fraser and Tsang, 2001) or Co: Writer (Johnston, 2016) to improve typing performance and reduce the physical demand on the user. Most of these prediction software has both a speech component and manual typing component to cater to a wider range of disabilities, and can be incorporated into existing word processors. Both Word Q and Co: Writer can interface with a wide range of word-processing applications including Microsoft Suite, Chrome amongst others. The word prediction itself for both systems are similar, with

Word Q using a vocabulary largely based on the user, and Co: Writer using topic-assigned vocabularies. Both systems have a speech output to aid in the typing process.

Apart from these commercialized systems, substantial research has also been conducted in identifying alternate word prediction methods and incorporating them into existing or novel AAC systems. As mentioned, many high-tech AAC devices exist however, in alignment with the project's focus, the literature review will be mainly considering the BCI system and specifically, word-typing based BCI such as the P300-Speller, RSVP Keyboard and other soft keyboards.

Ryan et al., (2010), examined the effects of predictive typing by comparing a standard P300-speller with a P300-speller that contained a predictive component (Word Q), studying particularly the trade-off between increased cognitive effort but reduced physical effort (decreased number of keystrokes). The study found that while output-character-per-minute (OCM) and time taken to complete a sentence was faster in the BCI with the predictive component, accuracy faltered presumed to be because of the general task difficulty associated with it. Increasing accuracy, however, could simply be achieved by more practice with the system.

Akram et al., (2013) initially suggested the idea of simply incorporating a custom-built dictionary into the BCI to provide word suggestions, before proposing a further modification to the BCI alphanumeric grid itself to better resemble the T9 interface commonly used in SMS messaging systems (Akram et al., 2015). The combination of both word suggestions and the T9 interface was able to reduce the typing time however, the available predictions are heavily constrained by the dictionary which only contains 1000 most-commonly used English words.

Aside from P300-Spellers, other typing-based BCIs exist and also attempt to improve the typing performance. Orhan et al., (2012) developed the RSVP (Rapid

Serial Visual Representation) which presents characters or symbols in the same place on screen making it cognitively easier on the user when selecting the characters (the selection is by EEG). The language model is used to increase the speed of selection by using the history of the characters typed to predict the next character that is flashed on the screen. This does increase the typing speed, but is restricted to character-by-character predictive typing rather than word or even sentence predictions.

DASHER (Figure 3.1) is an open-source software that provides a user interface to type text into a computer. It clusters all letters of the alphabet on one side of the interface and to type, the user navigates to the appropriate letter. Each chosen character has associated sub-boxes containing possible subsequent letters, where the size of the sub-box is proportional to the letter's probability given by the language model. Wills and Mackay (2006) proposed to incorporate this software in BCIs in an effort to improve its transfer rate by using the BCI signal to choose a letter in the DASHER interface. The main pitfall is that the DASHER interface is cognitively demanding, particularly if the signal is continuous however, is still relatively less than what is required of a P300-Speller.


*Figure 3.1 DASHER Interface Wills and Mackay (2006)*

Speier et al., (2011) also took the language model path to improve spelling and bit rate of BCIs, choosing to combine a stepwise linear discriminant analysis with Naïve Bayes classifier and a trigram model. The trigram model was used as

a prior for the Naïve Bayes classifier when determining the probability of each character. This approach was able to achieve significant improvement in accuracy and bit rates, with suggestions of expanding to word predictions as well as letter predictions.

General improvements to less technological-advanced typing systems such as generic soft keyboards are also important research areas. These modifications are all reliant on language models to provide prediction, but also attempt to use other factors such as gaze point to further enhance typing.

Mackenzie and Zhang (2008) used keyboard geometry, current fixation point of the user, previously inputted characters and a language model to build letter prediction (three most probable letters highlighted on a keyboard) and word prediction models. The combination of all these pieces of information means that more accurate predictions can be made particularly in letter prediction. For example, if after "th" the user is fixating on "d" but the most probable letter is "e", "e" will be highlighted instead. This method works well for letter prediction however is sensitive to the first few letters typed.

Goodman et al., (2002) make use of language models to reduce WER and word disambiguation particularly when a user clicks the boundary between two keys and the most probable key needs to be found on a soft keyboard. This allows for completely corrective keyboards. They found that the incorporation of language models significantly reduced the WER and increased the words per minute (WPM). Users also preferred the corrective keyboard as it was able to reduce the number of errors.

Numerous studies in both sections have highlighted the benefits of firstly, incorporating syntactic, semantic and topic information to standard language models and secondly, including word prediction algorithms in AAC devices. Maximum performance improvement of word prediction language models will evidently occur with a model that can exploit all three information sources to

provide predictions that are syntactically, semantically and topically accurate thus mimicking natural language. While such a model has not yet been successful implemented for word prediction, combinations such as syntactic-topic models and semantic-syntactic models have seen some success. Both models are viable however syntactic-semantic models may be more computationally efficient given its independence of LDA.

The studies also indicated the little focus placed on user-based language models specifically, predictions that are based on the user and their style of writing rather than on the language and style of some pre-determined documents or text. This project aims to address this research gap to design a user-based word prediction model capable of presenting predictions reflective of the user's interests, style of writing and the contextual nature of the task being undertaken.

# 4 Implementation

## 4.1 Programming Language

The program chosen for implementation was Python 3.5 due to its existing natural language toolkit (NLTK), cleaner syntax and most developed version of Python.

### *4.1.1 NLTK*

NLTK is a NLP toolkit available through Python. The package contains various corpora as well as text-processing libraries including frequency distribution, tokenization, corpus readers, stopping and stemming, lemmatisation, WordNet and word sense disambiguation (NLTK, 2015), all of which were used as part of developing the word prediction model. The various uses of these libraries in the application of word prediction are explained further in this chapter.

## 4.2 Methodology

The developed word prediction system can be split into three main sections – the user input, generation of predictions and the selection of the required predictions. The prediction module constitutes the bulk of the program including the developed word prediction model. Figure 4.1 illustrates the methodology implemented and the basic tasks of each module.

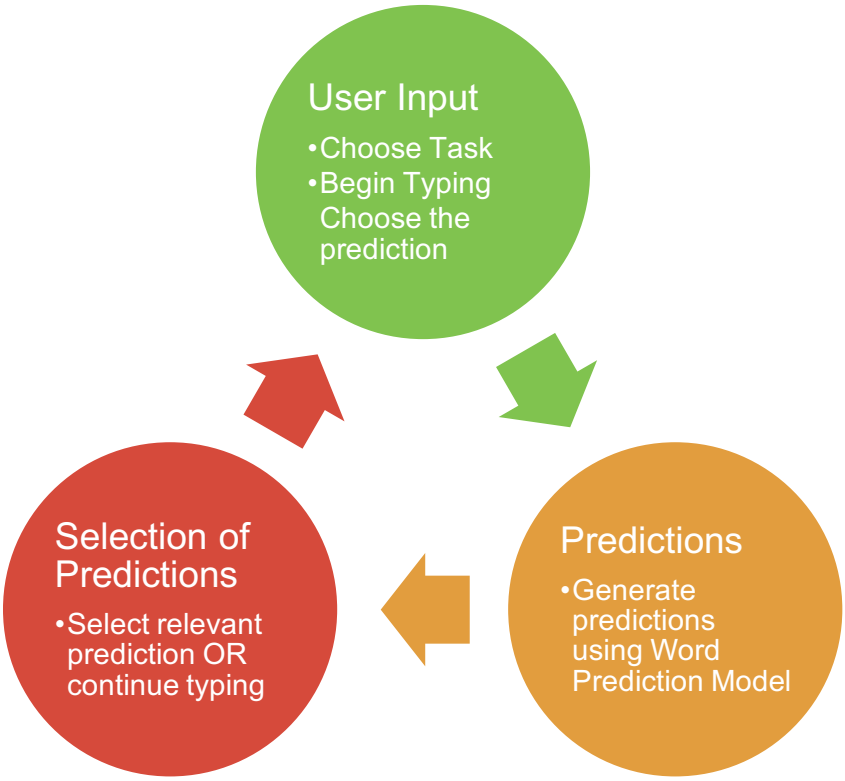


*Figure 4.1 Methodology of System*

### *4.2.1 User Input*

The user input has three main purposes:

- Identify the task that they wish to undertake
- Begin typing
- Choose the prediction

The task and context the user is undertaking dictates the task-derived user-based corpus (UBC) - in this case, the user has the option of choosing between writing an email or a document (task) and within email, a formal or informal email. It is assumed that the document task is a report, however in future versions this may include letters, literature reviews, resumes and so forth.

Once the task is identified, the user is prompted to begin typing. An un-buffered input is used such that predictions are automatically generated once the user presses the 'space' key, rather than waiting for the user to press 'enter' after each

word. Lastly, once the alphabetized prediction list is presented to the user, they can make a selection and the program will automatically append the word (and space) to the current document. Else, they can continue typing.

### *4.2.2 Predictions*

#### *4.2.2.1 Pre-Processing*

Pre-processing of the training sets for implementation of the word prediction model is required.

##### 4.2.2.1.1 Training Corpora

The purpose of the training corpora is for training the language model such that it can be tested during the testing phase. The training corpora consists of a UBC and the British National Corpus (BNC, British National Corpus, 2009). Two versions of the corpora were used to generate the predictions – an unfiltered corpus for syntactic-based predictions, and a filtered corpus for semantic-based predictions.

###### 4.2.2.1.1.1 User-Based Corpus

The aim of the UBC is to provide all relevant information about the user such that the predictions generated can be user-based and task-driven. These include but not limited to, emails, address books, read and written documents, webpages searched and saved etc. Each task has its own associated UBC – i.e., a corpus of emails, or a corpus of read and written documents and depending on the task being undertaken, the corresponding UBC is used. As the testing and results aim to validate the model as a 'proof-of-concept', and also be computationally efficient, the task-dependent UBCs for this model were predesigned consisting of personal emails (formal and informal), and saved PDF documents relating to this project and other university studies. The formal and informal email UBCs also contained part of the publicly available Enron corpus. All the emails and PDF documents were in text-file format, such that they could be read by the PlainTextCorpusReader from the NLTK package.

#### *4.2.2.1.1.1.1 Enron Email Dataset*

The Enron email dataset is a publicly available collection of emails collected and prepared by the CALO (A Cognitive Assistant that Learns and Organises). The dataset was first made public by the Federal Energy Regulatory Commission during their investigation of the company. The dataset itself contains emails from 150 employees of Enron (mostly senior management) and contains a total of 0.5 M messages (Enron Email Dataset, 2015). Of this, approximately 3000 emails (20 emails from each employee) were used for the UBC. This ensured that there was an equal number of Enron and personal emails in the UBC as well as equal number of emails taken from each employee. The classification of an informal vs. formal email was done subjectively, on the basis of whether the email was referring to professional work or personal work. An example of a formal email is shown in Figure 4.2 and an informal email in Figure 4.3. The informal and formal emails were filtered to only contain the subject and contents of the email as the focus of the software is to only aid in predicting words relevant to the body of the text.

```
Checklist for Review of Trading Contract Form (Canadian Form)

Alan, my quick impression is that the checklist format is identifying a
number of matters for discussion in a way that will be efficient.  What is
your impression?

Mark
```

*Figure 4.2 Enron Formal Email*

```
Hello!

Thank you for the leads, but I am going to stay here in "the land of grey winter",
until Todd is ready to move south.  That will hopefully be in 2 or 3 years.
```

*Figure 4.3 Enron Informal Email*

##### *4.2.2.1.1.1.2 Personal Email Dataset*

The personal email dataset consisted of approximately 3,000 emails (formal and informal). Like the Enron email dataset, the classification of formal and informal was subjective and dependent on whether the email was in reference to university work or personal work. The informal and formal emails were filtered to only contain the subject and contents of the email.

##### *4.2.2.1.1.1.3 Personal PDF Documents*

The PDF documents used in the UBC were based on the documents relating to this project (i.e., relevant papers, reviews) and other documents relating to other university studies (lecture notes, assignments). All papers were filtered to contain only the title, the authors and the main body of the text. Formulas which were unchanged in the text file were not filtered, as the predictions aim to be as accurate as possible given the context, i.e., if the user were writing a report with formulas used before, it would be beneficial for the model to be able to predict them. If, however, they were not recognised by the text file, the resulting replacement character was filtered.

#### 4.2.2.1.1.2 British National Corpus

The British National Corpus (BNC) is a 100 million text corpus containing written and spoken British English from a range of sources (British National Corpus, 2009). The BNC was acquired through Flinders University, and cleaned to remove the html tags. The code is shown in Appendix A.

### 4.2.2.1.2 Tokenisation

To simplify the text processing process, all of the corpora and test sets were tokenised. Tokenisation essentially splits a sentence into a series of tokens ranging from words, numbers or punctuation. The text was tokenised using the build in word tokeniser from NLTK. The following examples illustrates a raw text compared to a tokenised text.

*Example*
*Original:*

"This is a sample text that has not been tokenised!"

*Tokenised:*

'This' 'is' 'a' 'sample' 'text' 'that' 'has' 'not' 'been' 'tokenised' '!'

#### *4.2.2.2 Filtered and Unfiltered Corpus*

As mentioned, two versions of each (UBC, BNC) training corpora were used to predict both syntactic and semantic words. After the initial filtrations (removal of tags, date in emails etc.), the first version remained the same (unfiltered) while the second version implemented a second round of filters to remove common (stop) words leaving only content words. Lemmatising was also used to reduce inflectional forms of words i.e., 'liking' and 'liked' are both inflectional forms of the word 'like'.

##### 4.2.2.2.1 Stopwords

The removal of stopwords is achieved using NLTK, which provides its own default set of stopwords (Table 4.1)

*Table 4.1 NLTK Stopwords*

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| ourselves | hers | between | yourself | but | again | there | about | once | during |
| out | very | having | with | they | own | an | be | some | for |
| do | its | yours | such | into | of | most | itself | other | off |
| Is | s | am | or | who | as | from | him | each | the |
| themselves | until | below | are | we | these | your | his | through | Don’ |
| nor | me | were | her | more | himself | this | down | should | our |
| their | while | above | both | up | to | ours | had | she | all |
| no | When | at | any | before | them | same | and | been | have |
| in | will | on | does | yourselves | then | that | because | what | over |
| why | so | can | did | not | now | under | he | you | herself |
| has | just | where | too | only | myself | which | those | I | after |
| few | whom | t | being | if | theirs | my | against | a | by |
| doing | it | how | further | was | here | than | | | |

#### 4.2.2.2.2 Stemming

Both stemming and lemmatisation are methods to reduce inflectional forms of words, however stemming is a cruder process in which the end of the word is simply taken off whereas lemmatisation considers the morphology and existing vocabulary when returning the reduced form (Martin and Jurafsky, 2000). For example, for the word 'having', a stemmer may return 'hav' while a lemmatiser returns 'have'. Given that the semantic predictions are taken from the filtered corpus, it was important for the filtered words to have correct spelling, hence a lemmatiser was used instead of a stemmer. The lemmatiser used was the WordNet lemmatiser, which uses WordNet's in-built morphological processing tool for considering morphology and existing vocabulary. Given a string and a syntactic category (noun, verb etc.,) the WordNet morphological processor applies a set of 'rules of detachment' (Table 4.2) to the string and returns the base form of the word. If the word is an exception, WordNet searches through an exception list or simply returns the unchanged input word (Princeton University, 2010).

*Table 4.2 WordNet 'Rules of Detachment'*

| **Syntactic Category** | **Suffix** | **Ending** |
|---|---|---|
| Noun | "s" | "" |
| Noun | "ses" | "s" |
| Noun | "xes" | "x" |
| Noun | "zes" | "z" |
| Noun | "ches" | "ch" |
| Noun | "shes" | "sh" |
| Noun | "men" | "man" |
| Noun | "ies" | "y" |
| Verb | "s" | "" |
| Verb | "ies" | "y" |
| Verb | "es" | "e" |
| Verb | "es" | "" |
| Verb | "ed" | "e" |
| Verb | "ed" | "" |
| Verb | "ing" | "e" |
| Verb | "ing" | "" |
| Adjective | "er" | "" |
| Adjective | "est" | "" |
| Adjective | "er" | "e" |
| Adjective | "est" | "e" |

#### *4.2.2.3 Word Prediction Model*

As mentioned in the Background section, the three commonly used *n*-gram models for assigning probabilities to words are the unigram model (unconditioned), the bigram model (conditioned on the previous word) and the trigram model (conditioned on the previous two words). While the accuracy of the model increases with higher *n*-grams, this also increases the computational

inefficiency, and therefore typically bigram or trigram models are used as they ensure accuracy while still maintaining computational efficiency. In this case, the bigram model was used as it optimised the trade-off between computational efficiency and accuracy.

The prediction themselves are based on the bigram counts, or a conditional frequency distribution (CFD) of all the bigrams in the given corpus. The distribution contains the bigram counts for all bigrams in the corpus, and the most frequently occurring outcome word following the input or context word is used as a potential prediction. This was implemented using the probability module from NLTK.

### *Example*

Using the training sentence "*the quick brown fox jumps over the lazy dog",* Table 4.3 illustrates the CFD of bigram counts for the training sentence with the predictions highlighted. Note: the rows of the table depict the *context* words and the columns of table show the *outcome* or prediction words.

*Table 4.3 Conditional Frequency Distribution of Training Sentence*

| **CFD** | **Brown** | **Dog** | **Fox** | **Jumps** | **Lazy** | **Over** | **Quick** | **The** |
|---|---|---|---|---|---|---|---|---|
| **Brown** | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| **Fox** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| **Jumps** | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| **Lazy** | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| **Over** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| **Quick** | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **The** | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |

To decrease computational time, the CFD of the BNC and UBC (both filtered and unfiltered versions) were saved in a dictionary format (Appendix B) using the

pickle module such that it could be re-opened easily rather than having to generate bigram counts in the corpus for every input word.

#### 4.2.2.3.1 Prediction Sources

The predictions are taken from three different sources where the first source is an unfiltered corpus, the second is a filtered (no stopwords and stemmed words) corpus and the third is WordNet (Princeton University, 2010). The purpose of using the first two sources is to predict both syntactic (unfiltered) and semantic (filtered) words. The syntactic-based words are however, a crude reflection of true syntax given the underpinning assumption that the syntax in the corpus is correct. The semantically-related words are taken from the filtered corpus with the assumption that the removal of stop-words and inflectional words leaves content words which are more semantically related to each other.

Finally, the WordNet predictions are also semantic-driven predictions, and synonyms of the input word. WordNet is an online lexical database in which words (nouns, verbs, adjective and adverbs) are grouped into synsets which represent a particular concept i.e., the 'animal' synset corresponds to words relating to animals. Evidently, words under a particular synset are all synonymous to each other (Princeton University, 2010).

To increase the accuracy of the WordNet predictions, Word-Sense Disambiguation (WSD) methods were used to find the 'sense' of the word and based on the closest related-synset, extract the 'lemmas' or words, in that synset (i.e., the synonyms) and add it to the overall prediction list.

There have been several WSD methods proposed over the years (Yang and Powers 2005, Yang and Powers 2006, Powers 1997, Huang and Powers 2001) of which the popular Lesk algorithm was used for this model. Two main versions of the Lesk algorithm exist namely the original (Lesk 1986) and the adapted Lesk algorithm (Banerjee and Pederson, 2002), both of which are available through

the Lesk NLTK module. The basic Lesk algorithm works by selecting the word-sense (glosses) whose definition has the most overlap (highest number of common words) with the context (previously written words). The adapted Lesk algorithm used in model uses WordNet as the basis for the sense definitions and outputs the corresponding synset given the context. Therefore, the more words in the context, the more likelihood of overlapping common words between the sentence and synset definition and the more accurately returned synset. Given that WordNet predictions are only generated for nouns, verbs, adjectives or adverbs, this constraint was added in the system to reduce computation time**.** The WordNet predictions are generated independent of the syntactic and semantic predictions and if they exist, are simply appended to the existing prediction list.

#### 4.2.2.3.2 Model

The model designed is based on the back-off SLM adaptation method, also known as the fill-up technique (see Background). The generalised equation for a back-off SLM adaptation model is given by Equation 2.15 (page 18), where $\Pr(A)$ represents the SLM based on the dynamic corpus A, and $\Pr(B)$ is the SLM based on the static background corpus B.

Instead of having two corpora however, the designed model uses three corpora where the current document and the UBC are the dynamic corpora, and the BNC is the static background corpus B. Moreover, two versions of each corpus exist, and the bigram predictions are based on *both* corpora. That is, the bigram predictions from the overall current document, UBC and BNC corpora incorporate all the predictions from the filtered and unfiltered corpora (repeat predictions are discarded). Since the predictions from each filtered and unfiltered corpora are mutually independent i.e., the predictions generated from the filtered corpus are independent of the predictions generated from the unfiltered corpus, the combination is based on the multiplication rule,

$$P(A \text{ and } B) = P(A) * P(B) \tag{4.1}$$

The overall back-off model equation therefore becomes,

$$\Pr(w_q|h_q) = \begin{cases} Pr_{A11}(w_q|h_q) \times Pr_{A12}(w_q|h_q) & if\ C_{A11}(h_q w_q)\ or\ C_{A12}(h_q w_q) \geq T \\ \beta Pr_{A21}(w_q|h_q) \times Pr_{A22}(w_q|h_q) & if\ C_{A21}(h_q w_q)\ or\ C_{A22}(h_q w_q) \geq T \\ \beta Pr_{B11}(w_q|h_q) \times Pr_{B12}(w_q|h_q) & otherwise \end{cases} \quad (4.2)$$

Table 4.4 shows the relevant variables and their representation. The implementation of the model is shown in Appendix C. Note: The WordNet predictions are generated independently of the corpora-based predictions, and therefore not included in the equation.

*Table 4.4 Variables and Representations*

| **Variables** | **Representation** |
|---|---|
| $\boldsymbol{A_{11}}$ | Unfiltered current document corpus |
| $\boldsymbol{A_{12}}$ | Filtered current document corpus |
| $\boldsymbol{A_{21}}$ | Unfiltered UBC |
| $\boldsymbol{A_{22}}$ | Filtered UBC |
| $\boldsymbol{B_{11}}$ | Unfiltered BNC |
| $\boldsymbol{B_{12}}$ | Filtered BNC |
| $\boldsymbol{T}$ | 1 (bigram count must be at least 1 otherwise back-off to next corpus) |
| $\beta$ | 1, back-off coefficient |

### *4.2.3 Selection*

The predictions listed on the final UCI are subject to screen availability and therefore to reflect this constraint, the prediction lists only the three most common predictions (i.e., those with the highest probability) from each source. This also ensures that the system remains computationally efficient. Once the list of predicted words is presented to the user, they are given two options –

choose a word from the list or continue typing. If they choose the former, the chosen word is appended to the existing sentence, and the next set of predictions are generated based on the newly appended word. If the prediction list does not contain the intended word, the user can continue to type and the next set of predictions will be based on the next word the user types. The process is repeated until the user terminates the program.

*Example*

*Email:*
*"Good Morning, meeting at 9am?"*

The user is first asked to respond to the question "*Would you like to write a document or email?"*

In this case, the user would type "*email"*. Note: the system is case-sensitive.

Pursuant to this, the user is further asked "*Would you like to write a formal or informal email?"* to identify the relevant UBC.

In this case, the user would type "*formal"*.

The user is then prompted to begin typing.

Following the first word "*Good"* the system first appends the word to a running document, before loading predictions. As previously mentioned, the word prediction system works as a 'back-off' that is, it first queries the current document for a prediction, then the UBC and finally the BNC. The querying itself is done by first identifying whether the current word, i.e. "*Good"* is in the current document, the UBC or the BNC. Once the current word has been found in either of the three corpora, then the system organises all the words in the corresponding corpus in bigrams such that the CFD method can be used to

generate the predictions. As mentioned, the prediction list contains both syntactic and semantic predictions, and as such the system takes the top 3 syntactic and semantic predictions, creates a new final prediction list and presents this list to the user. Using this list, the user is then asked to "…*Choose a prediction by typing 1-9 else 10 to continue typing".*

In this example, the predictions listed are as follows –

1. Morning
2. Luck
3. morning
4. good
5. well

The user would type "1" corresponding to the prediction "*Morning".* The system then automatically appends this predicted word to the running document and generates predictions for the predicted word (*"Morning").* This process continues until the last word has been either typed or predicted, and the user exits the system.

# 5 Experiments

Four baseline versions of the prediction model were tested and compared to a Baseline 0 model. The purpose of the baselines is to identify which is the most accurate and beneficial such that future versions can improve upon its performance.

## 5.1 Baseline 0

The Baseline 0 model assumes all words are of equal probability, and therefore provides *all* the words in the corpus as predictions.

## 5.2 Baseline 1

Baseline 1 is the most basic version of the model in which the predictions are based on unigram predictions. That is, the predictions listed are independent of the word typed and are based purely on the frequency of words in the corpus. For example, if the frequency distribution for the UBC lists 'the', 'and', 'a' and 'for' as the most frequently used words, they will be listed as the predictions for *any* new test word. The UBC used contains all emails (Enron and personal) and PDF documents i.e., it does not take task into consideration and therefore a generic UBC is used rather than a task-derived UBC.

## 5.3 Baseline 2

The second version also uses the generic UBC containing all emails and PDF documents, but instead of using unigram predictions, it uses bigram predictions (i.e., takes the input word into consideration).

## 5.4 Baseline 3

The third version also uses bigram predictions but instead of using the generic UBC, the task-derived UBC is used instead. For example, if the user is writing an informal email, then the UBC would be the informal email corpus.

## 5.5 Baseline 4

The fourth and final version also uses bigram predictions and a task-driven UBC, but also includes WordNet predictions.

Table 5.1 summarises the key differences between the five baselines.

*Table 5.1 Key Differences Between Baselines*

| **Baselines** | **Differences** |
|---|---|
| Baseline 0 | All words as predictions |
| Baseline 1 | Unigram predictions from generic UBC |
| Baseline 2 | Bigram predictions from generic UBC |
| Baseline 3 | Bigram predictions from task-driven UBC |
| Baseline 4 | Bigram predictions from task-driven UBC + WordNet predictions |

The primary aims of the test were to:

- Identify how well each model is capable of word prediction
- Evaluate the advantages and disadvantages of using a task-driven UBC vs. a generic UBC
- Determine if the inclusion of WordNet predictions is beneficial to the model

## 5.6 Test Set

The end user for this model is a person who is unable to type and therefore needs to type using alternative means such as their brain signals. Such persons are unlikely to begin using this interface for typing large documents. Rather, they are more likely to use the interface to write brief messages or emails. As such, preliminary tests conducted to test the feasibility and accuracy of the model were on a test set consisting of a 100 relatively short emails (approximately 200 characters), from both personal emails and the Enron database, as the training

UBC contains both sources of emails. Of the 100, 50 emails were informal emails and 50 were formal emails of which 25 were personal formal and informal emails and 25 were informal and formal emails from the Enron database. All the emails were filtered to contain only the subject heading and the contents of the message, and were subjectively classified as informal or formal based on whether they related to professional or personal work. The test set is "held out" data, and therefore mutually exclusive from the training set.

Figure 5.1 illustrates the breakdown of the test set. Depending on the results obtained, the tests can be extended to test larger emails or even office-related documents i.e., reports.

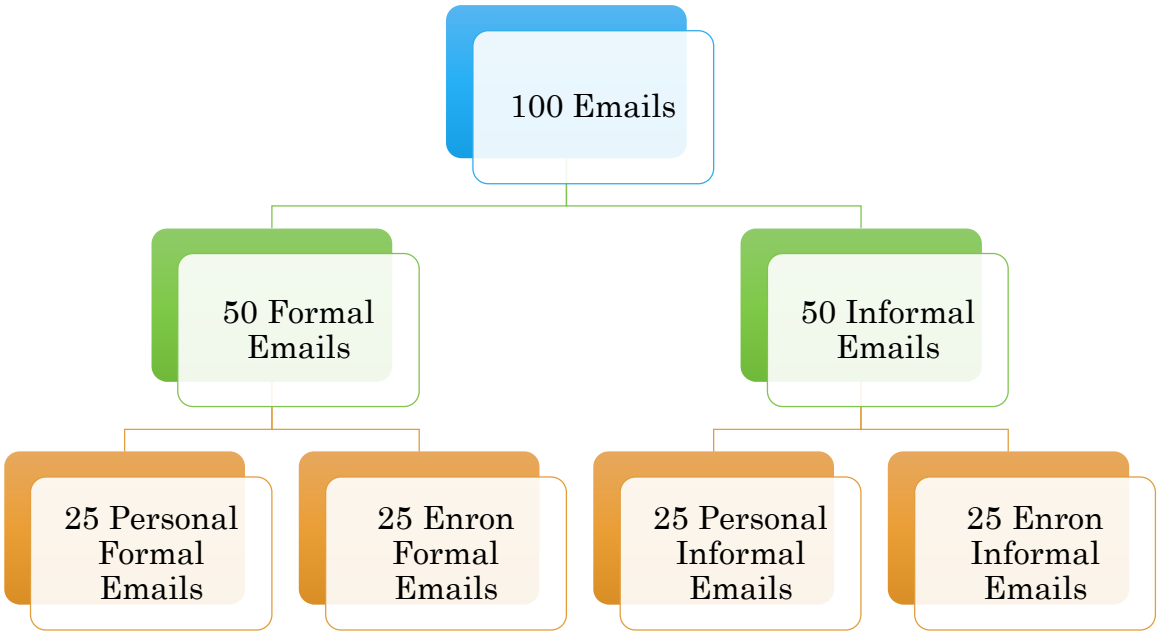


*Figure 5.1 Test set Breakdown*

## 5.7 Testing Implementation

### *5.7.1 Evaluation Metrics*

The two main evaluation metrics used for assessing the word prediction system were KSR and the keystrokes typed (KT). While the latter is not a mainstream evaluation metric for word prediction, KSR is an established word-prediction assessor, particularly for AAC devices and tend to be calculated using a human

subject. Given the timeframe and budget considerations of this project, an automated system was used rather than human subjects. As such, the results generated from this project cannot strictly be compared with those used in other studies as the assessment methods themselves differ. It is anticipated that in future versions particularly with the inclusion of the prediction system in the UCI, the KSR will be re-evaluated and used in comparison with existing studies.

#### *5.7.1.1 Keystroke Savings Rate*

The primary evaluation metric was the KSR (see Background section) which evaluates the keystrokes saved with the prediction tool. The higher the KSR, the better the prediction model. Calculation of the KSR is shown in Equation 2.17 (page 20). From the equation, the total entered characters are the total characters in each email without prediction while the keys pressed is the number of keys pressed when used in conjunction with the prediction tool.

#### *5.7.1.2 Normalised Keystrokes Typed*

A secondary evaluation metric was keystrokes typed (KT), which is equivalent to the number of keys pressed with and without the word prediction model, normalised by the total number of characters. That is,

$$Normalised\ KT = \frac{Total\ KT}{total\ characters}$$

The lower the KT, the more accurate the predictions generated and the better the model.

### *5.7.2 Testing*

#### *5.7.2.1 Total Entered Characters*

As mentioned, the total entered characters are a measure of how many characters are typed if there is no prediction i.e., the total number of characters

in each email. To streamline the process, the calculation was done automatically by tokenising the text and adding the length of each tokenised element. Note in this case, white space is not included in the total entered characters as the system automatically appends the whitespace once the user either selects a prediction or continues typing. The example below illustrates the process.

Example

Sentence: “I am going to the shops”
Tokenized: “I” “am” “going” “to” “the” “shops”
Total Keystrokes: 1 + 2 + 5 + 2 + 3 + 5 = 18

Additionally, the KSR is designed to only identify the accuracy of the predictions and therefore does not include secondary keystrokes such as enter (for appending predictions).

#### *5.7.2.2 Keys Pressed*

The keys pressed evaluates the number of keys pressed *with* word prediction. This again, was an automated process. For each word in the test email, the prediction model is used to predict the next word. If the model predicts the next word correctly, the number of keystrokes remains unchanged however, if the predictions are incorrect, then the number of keystrokes is increased by the length of unpredicted word (i.e., the cost of an error). This process is illustrated in Figure 5.2.

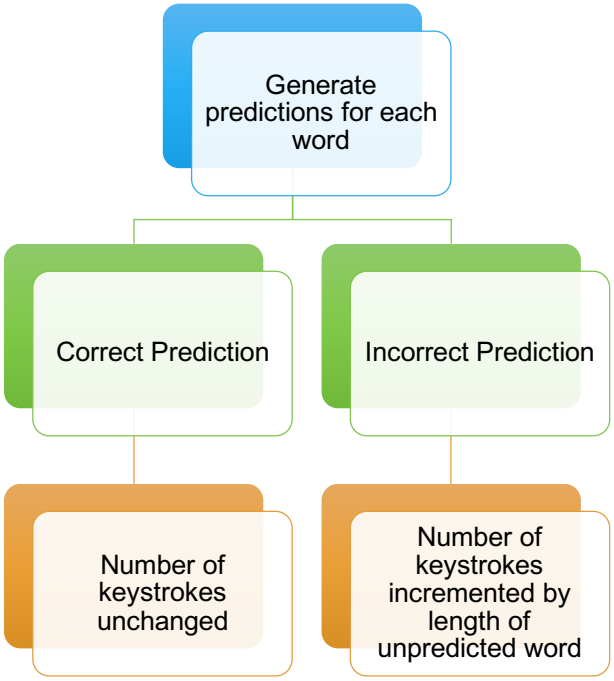


*Figure 5.2 Automated Keys Pressed Process*

# 6 Results

The results were generated on the four test sets. As Baseline 0 does not yield any predictions, KSR is 0 and therefore is difficult to accurately compare against the remaining Baselines. As such, the evaluation metric used to compare Baseline 0 to the other Baselines was total KT (Figures 6.1–6.4) whereas the evaluation metric to compare Baselines 1-4 (Figures 6.5-6.8) was KSR. In both cases, KT and KSR were calculated across *all* the emails in each test set. Table 6.1 shows the normalised, average KT values for each test set (along with the standard error) and Table 6.2 shows the total KSR values for each test set (along with the standard error). Note: as aforementioned, the KT values are normalised with respect to the total characters in each email therefore Baseline 0 is 1. Individual KSR and normalised KT values (with standard error) for each email in the test sets are shown in Appendices D and E respectively.

## 6.1 Standard Error

To account for statistical chance, the standard error was calculated for all the results, and represented in the graphs by the standard error bars. These error bars represent how precise the data is by accounting for statistical chance. If the error bars overlap, the findings are statistically insignificant where as if the error bars do not overlap, the findings are statistically significant. The equation for standard error is,

$$SE_{\bar{x}} = \frac{\sigma}{\sqrt{N}} \tag{6.1}$$

where

$\sigma$ = standard deviation

$N$ = number of observations in sample

(6.2)

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

where

$N$ = number of observations in sample

$\mu$ = sample mean

$x_i$ = sample values

In this case, $N = 25$ (25 emails). The standard deviation and standard error were computed using Excel.

Note: Baseline 0 does not take into account standard error as it is the true value.

## 6.2 T-Test

To further add credibility to the findings, a series of t-tests were conducted on the results to assess statistical significance. The null hypothesis states that "*there is no statistical significance between the two test sets*". If the absolute t-stat value is *greater* than the t-critical value, the null hypothesis is rejected. The KSR T-test results for the four test sets are shown in Tables 6.3-6.6 and the KT T-test results are shown in Tables 6.7-6.10.

*Table 6.1 Average KT for testset of 25 emails across 5 Baselines. The table also includes the standard error calculated on the 25 sample normalised KT values.*

| KT Total | Baseline 0 | Baseline 1 | Baseline 2 | Baseline 3 | Baseline 4 |
|---|---|---|---|---|---|
| **Enron Informal** | 1 | 0.954±0.484 | 0.864±1.261 | 0.843±1.307 | 0.839±1.311 |
| **Personal Informal** | 1 | 0.949±0.662 | 0.866±1.437 | 0.851±1.689 | 0.850±1.70 |
| **Personal Formal** | 1 | 0.955±0.609 | 0.863±1.938 | 0.851±2.065 | 0.844±2.082 |
| **Enron Formal** | 1 | 0.958±0.581 | 0.874±1.405 | 0.867±1.354 | 0.869+1.317 |

*Table 6.2 Average KSR for testset of 25 emails across 5 Baselines. The table also includes the standard error calculated on the 25 sample KSR values.*

| KSR Total | Baseline 0 (%) | Baseline 1 (%) | Baseline 2 (%) | Baseline 3 (%) | Baseline 4 (%) |
|---|---|---|---|---|---|
| **Enron Informal** | 0 | 4.960±0.484 | 13.941±1.261 | 15.764±1.307 | 16.139±1.311 |
| **Personal Informal** | 0 | 5.109±0.662 | 15.464±1.437 | 15.636±1.687 | 14.739±1.700 |
| **Enron** | 0 | 5.487±0.609 | 15.647±1.938 | 16.551±2.065 | 17.124±2.082 |

| | | | | | |
|---|---|---|---|---|---|
| **Formal Enron Informal** | 0 | 4.538±0.581 | 14.214±1.405 | 15.159±1.355 | 14.560±1.317 |

*Table 6.3 KSR T-Test Results for Enron Informal Emails*

| **Enron Informal (KSR)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | -9.486 | -10.821 | -12.017 | -12.3 | -6.706 | -7.976 | -8.257 | -1.133 | -1.365 | -0.230 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.039 | 2.042 | 2.042 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.4 KSR T-Test Results for Personal Informal Emails*

| **Personal Informal (KSR)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | -7.698 | -9.313 | -8.838 | -8.818 | -5.234 | -5.417 | -5.422 | -0.698 | -0.725 | -0.027 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.039 | 2.039 | 2.039 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.5 KSR T-Test Results for Enron Formal Emails*

| **Enron Formal (KSR)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | -7.298 | -8.998 | -9.841 | -9.952 | -5.528 | -6.170 | -6.161 | -0.355 | -0.241 | 0.121 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.035 | 2.035 | 2.035 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.6 KSR T-Test Results for Personal Formal Emails*

| **Personal Formal (KSR)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | -7.448 | -7.052 | -7.205 | -7.470 | -4.493 | -4.803 | -5.077 | -0.429 | -0.662 | -0.229 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.039 | 2.042 | 2.042 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.7 KT T-Test Results for Enron Informal Emails*

| **Enron Informal (KT)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | 9.485 | 10.821 | 12.017 | 12.3 | 6.706 | 7.976 | 8.257 | 1.133 | 1.365 | 0.230 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.039 | 2.042 | 2.042 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.8 KT T-Test Results for Personal Informal Emails*

| **Personal Informal (KT)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | 7.698 | 9.313 | 8.838 | 8.818 | 5.234 | 5.417 | 5.422 | 0.698 | 0.725 | 0.027 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.039 | 2.039 | 2.039 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.9 KT T-Test Results for Enron Formal Emails*

| **Enron Formal (KT)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | 7.298 | 8.998 | 9.841 | 9.952 | 5.528 | 6.170 | 6.161 | 0.355 | 0.241 | -0.121 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.035 | 2.035 | 2.035 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

*Table 6.10 KT T-Test Results for Personal Formal Emails*

| **Personal Formal (KT)** | **B0 vs B1** | **B0 vs B2** | **B0 vs B3** | **B0 vs B4** | **B1 vs B2** | **B1 vs B3** | **B1 vs B4** | **B2 vs B3** | **B2 vs B4** | **B3 vs B4** |
|---|---|---|---|---|---|---|---|---|---|---|
| **T-stat** | 7.448 | 7.052 | 7.205 | 7.470 | 4.493 | 4.803 | 5.077 | 0.429 | 0.662 | 0.229 |
| **T-critical** | 2.064 | 2.064 | 2.064 | 2.064 | 2.039 | 2.042 | 2.042 | 2.010 | 2.010 | 2.010 |
| **Statistically Significant** | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No | No |

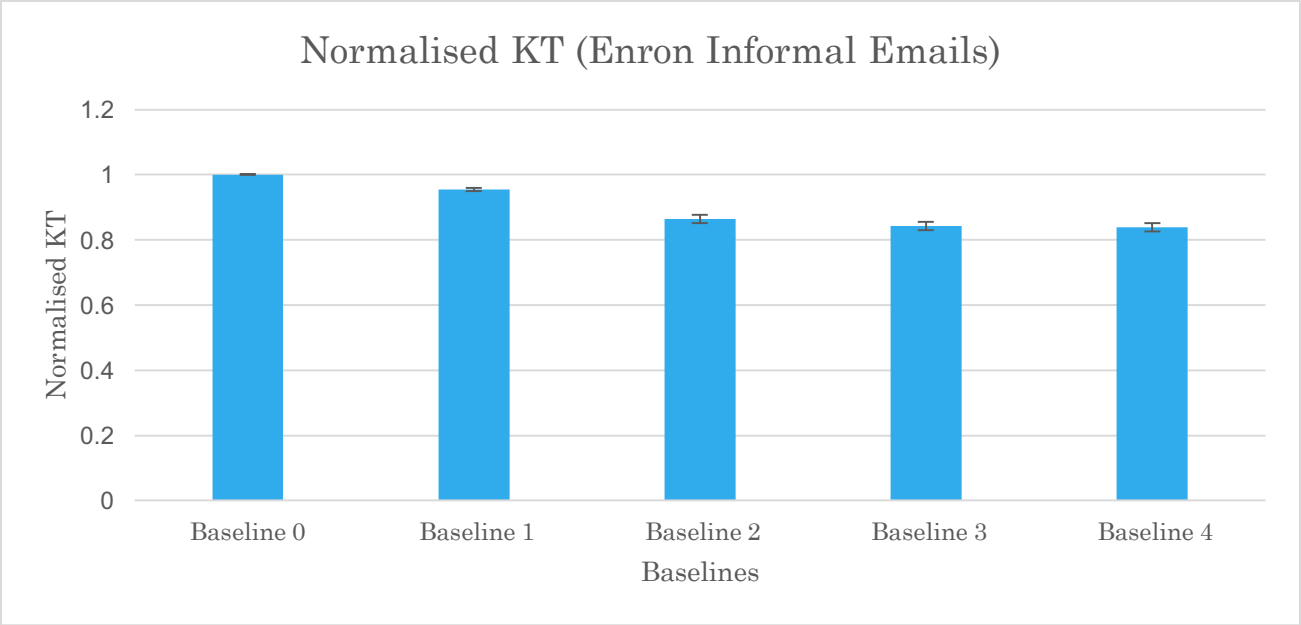


*Figure 6.1 Normalised KT for Enron Informal Emails*

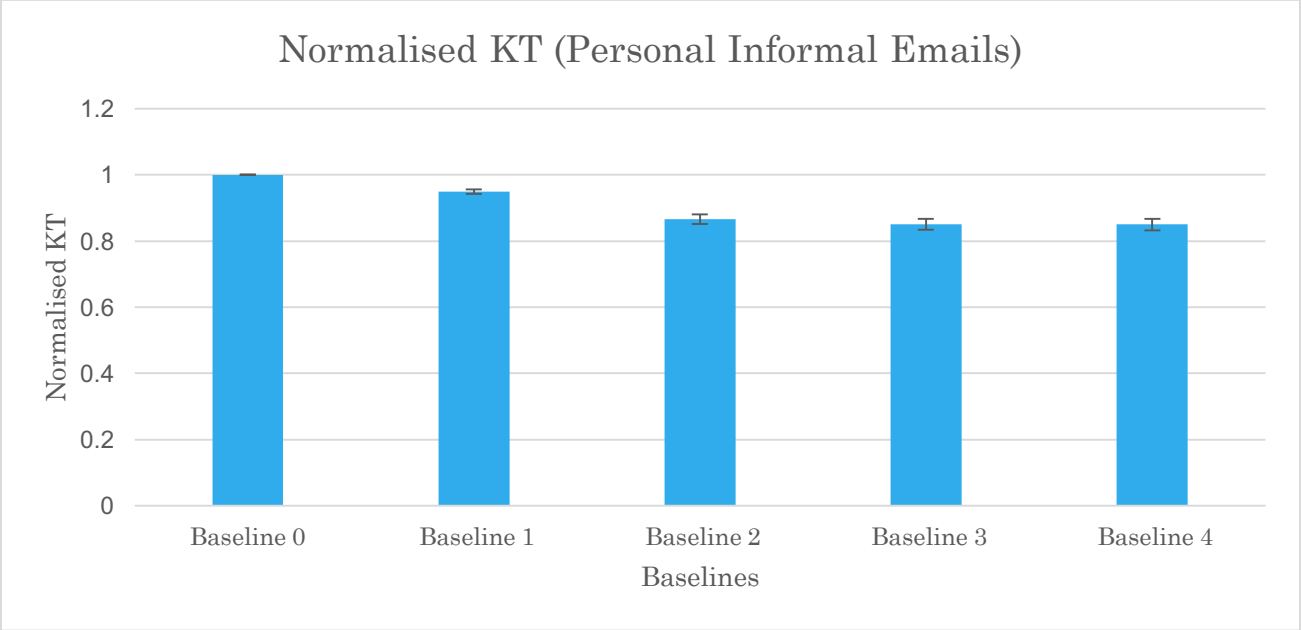


*Figure 6.2 Normalisedl KT for Personal Informal Emails*

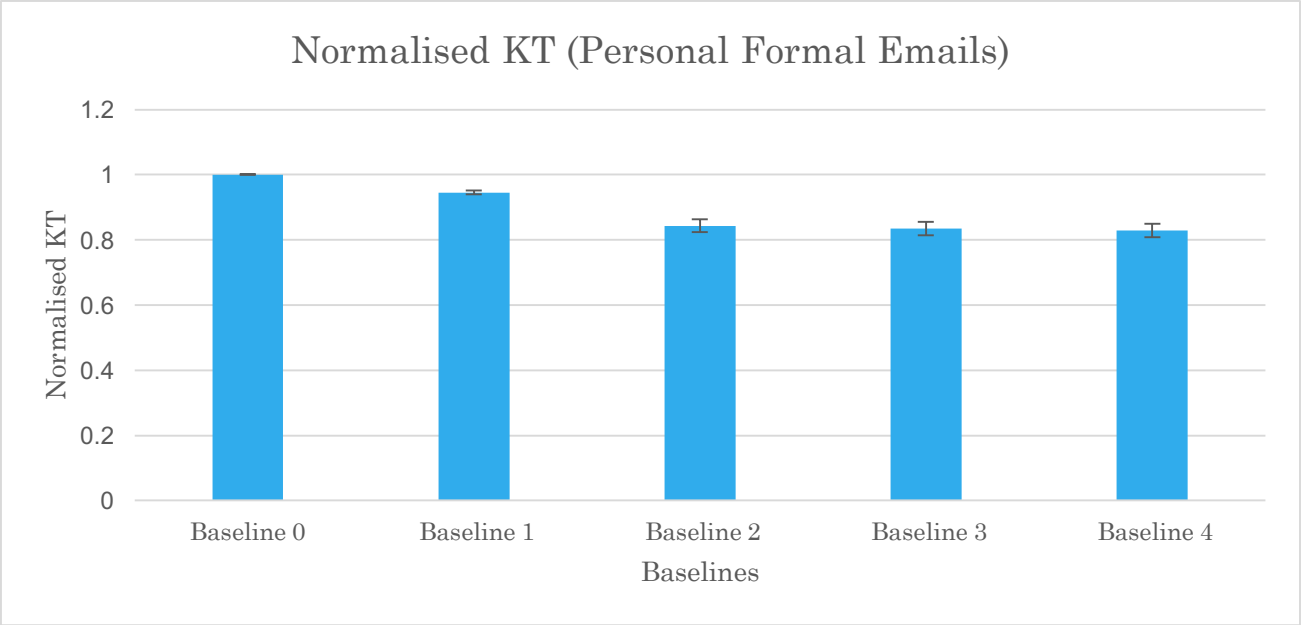


*Figure 6.3 Normalised KT for Personal Formal Emails*

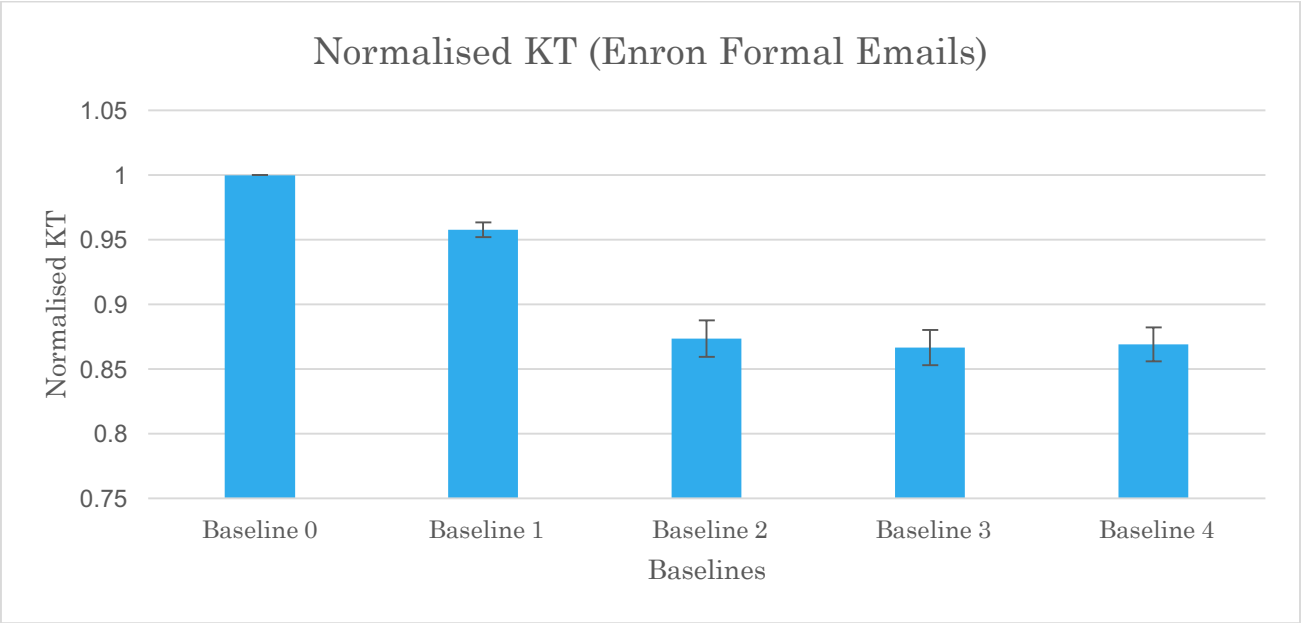


*Figure 6.4 Normalised KT for Enron Formal Emails*

Taking statistical significance into consideration, Baselines 1-4 are all able to reduce the number of characters typed with respect to Baseline 0 (equivalent to total characters in test set). In comparison to Baseline 1, Baselines 2-4 all fare better however in comparison with each other, there is no statistical difference.

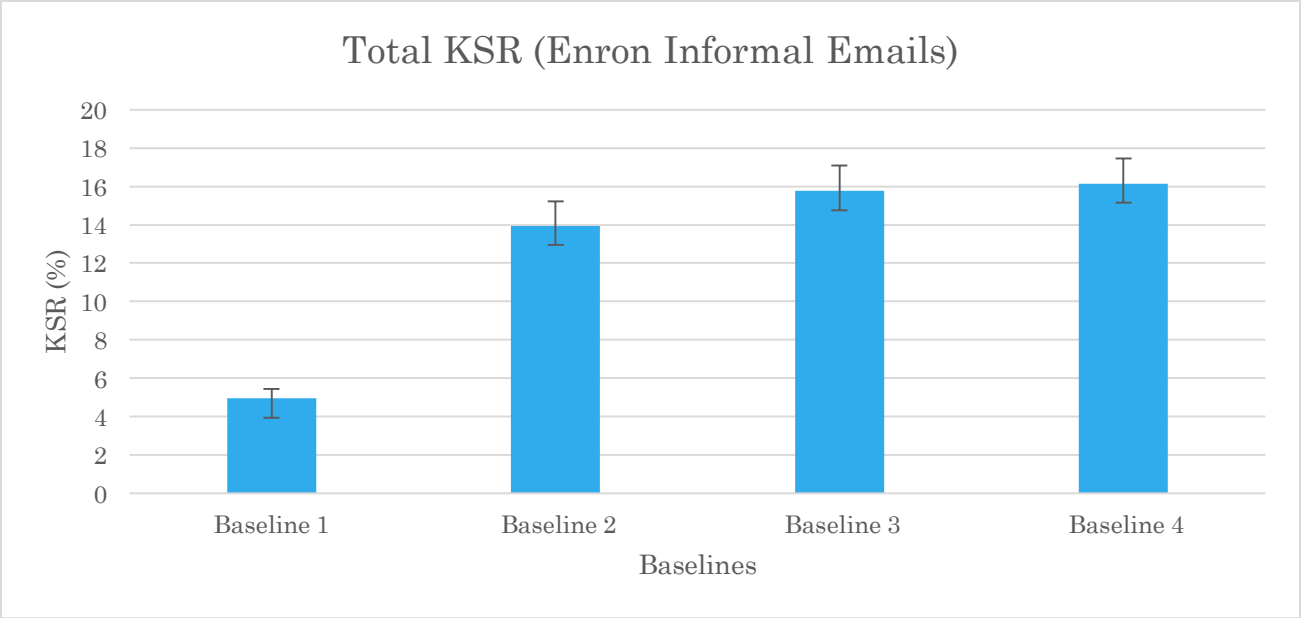


*Figure 6.5 Total KSR for Enron Informal Emails*

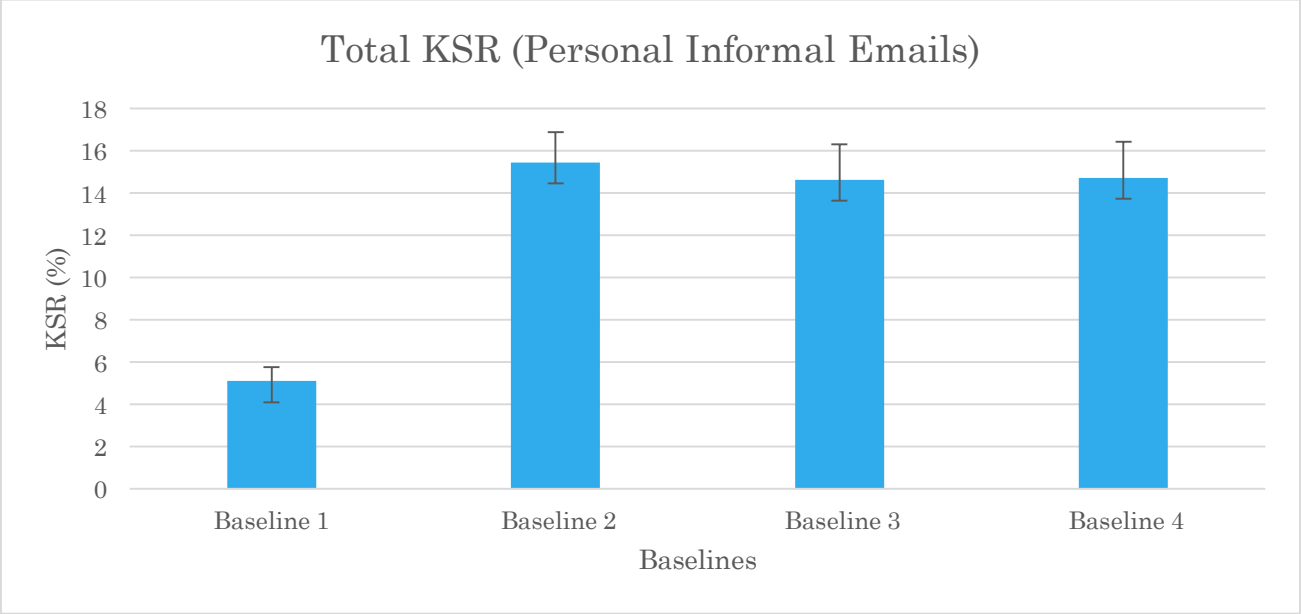


*Figure 6.6 Total KSR for Personal Informal Emails*

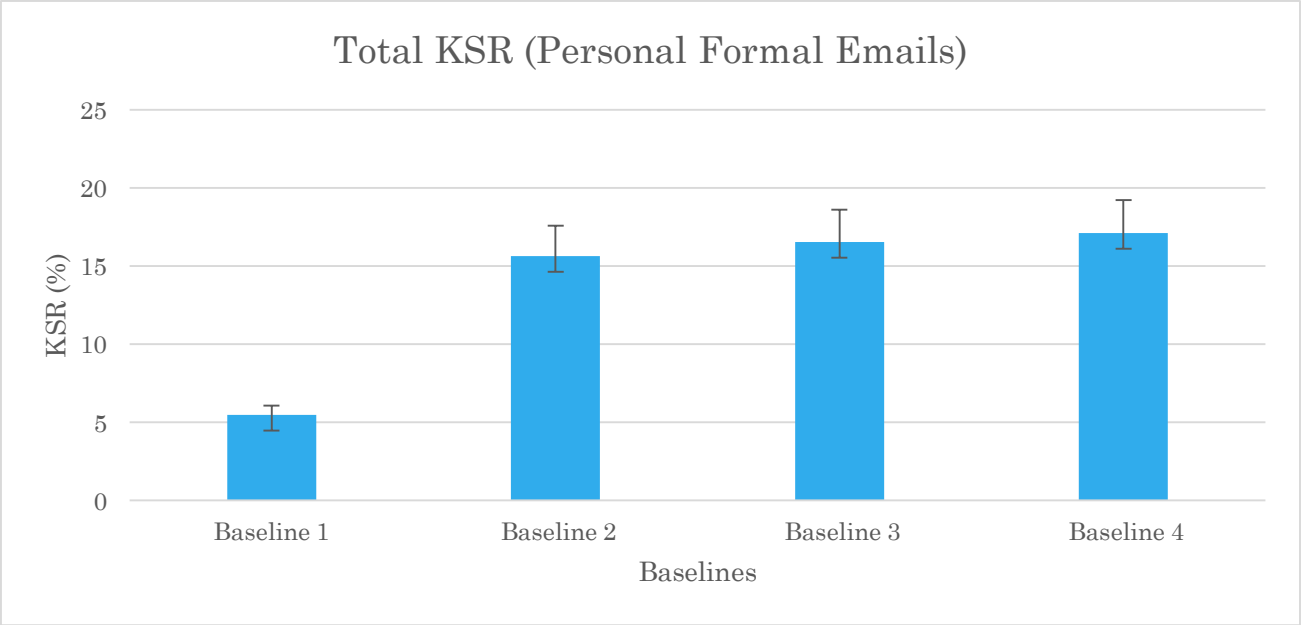


*Figure 6.7 Total KSR for Personal Formal Emails*

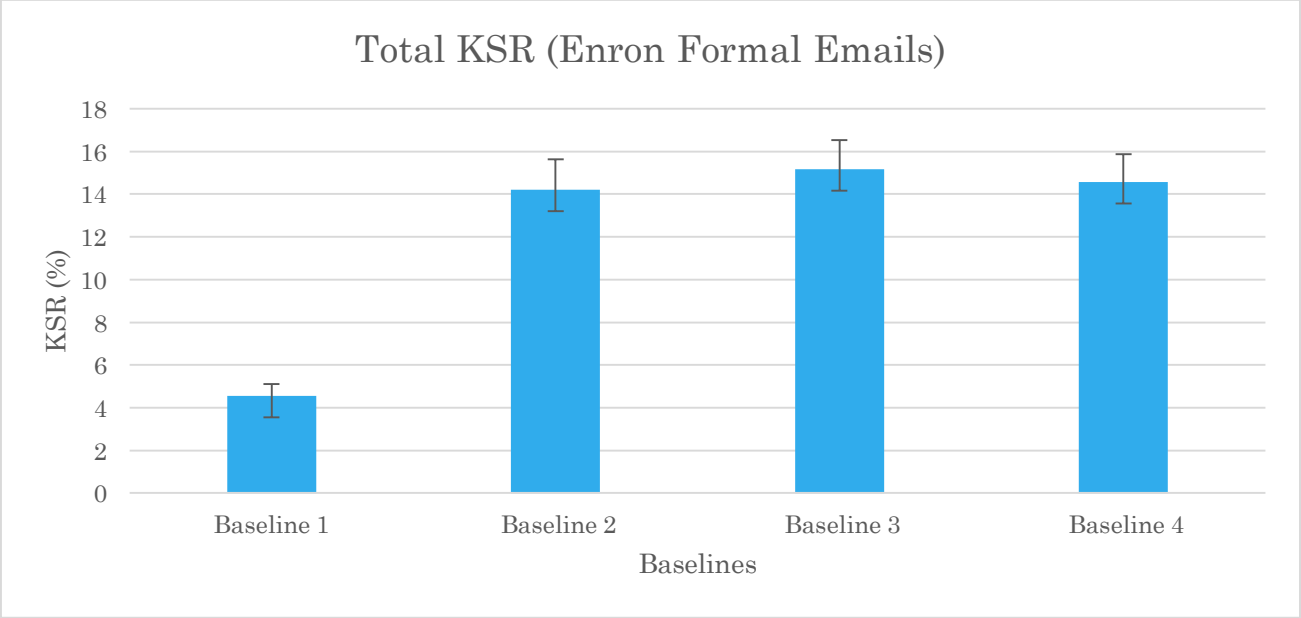


*Figure 6.8 Total KSR for Enron Formal Emails*

Considering statistical significance, Baselines 2-4 are all better than Baseline 1 however, are all statistically insignificant when compared with each other. Note: the KSR for Baseline 0 is zero, and therefore all Baselines are statistically significantly better than Baseline 0.

The following graphs illustrate the individual KSR for each email in each test set (Figures 6.9-6.12) and the corresponding distribution of email length (Figures 6.13-6.16) to identify a potential correlation between email length and KSR.

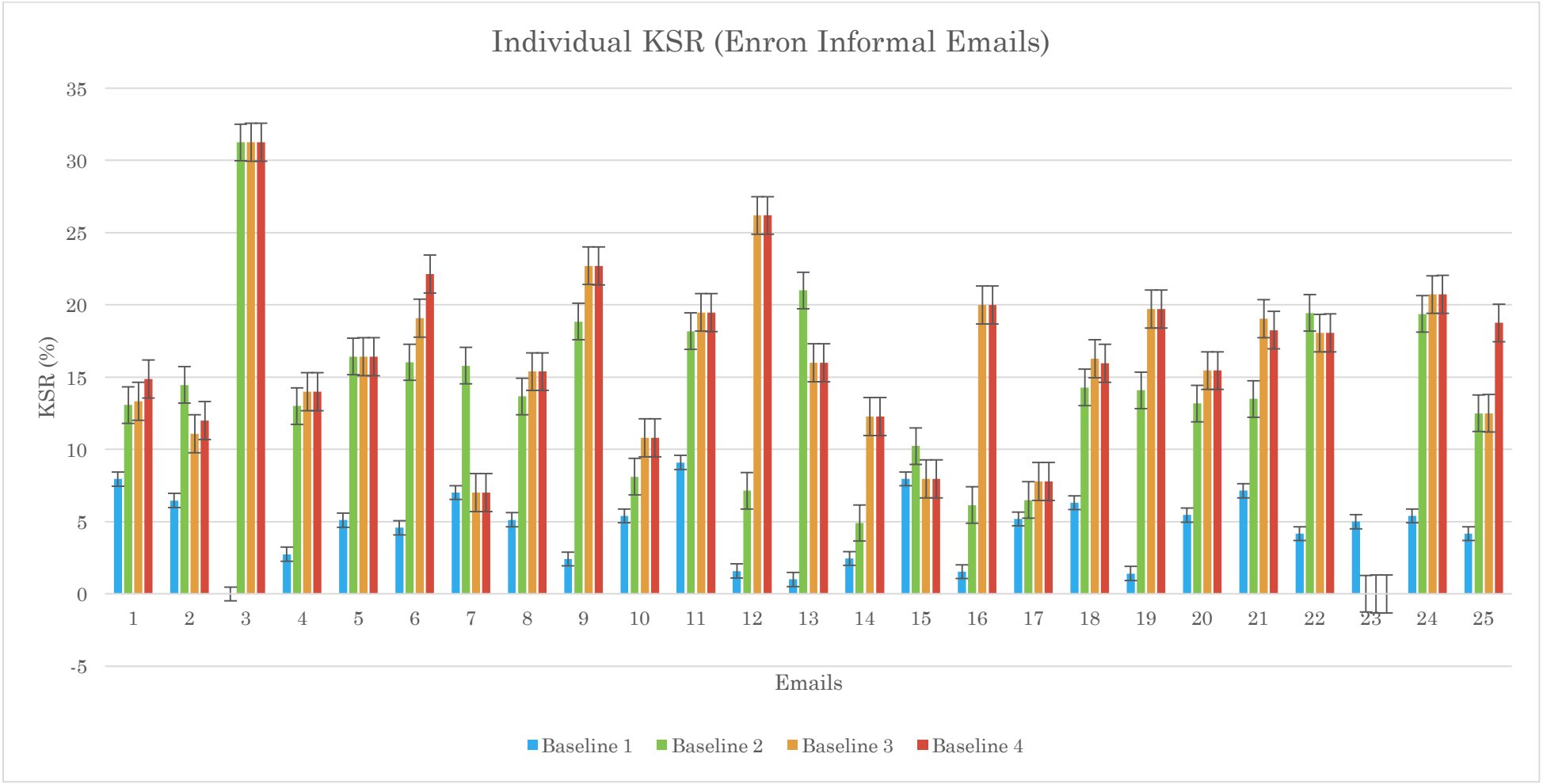


*Figure 6.9 Individual KSR for Enron Informal Emails*

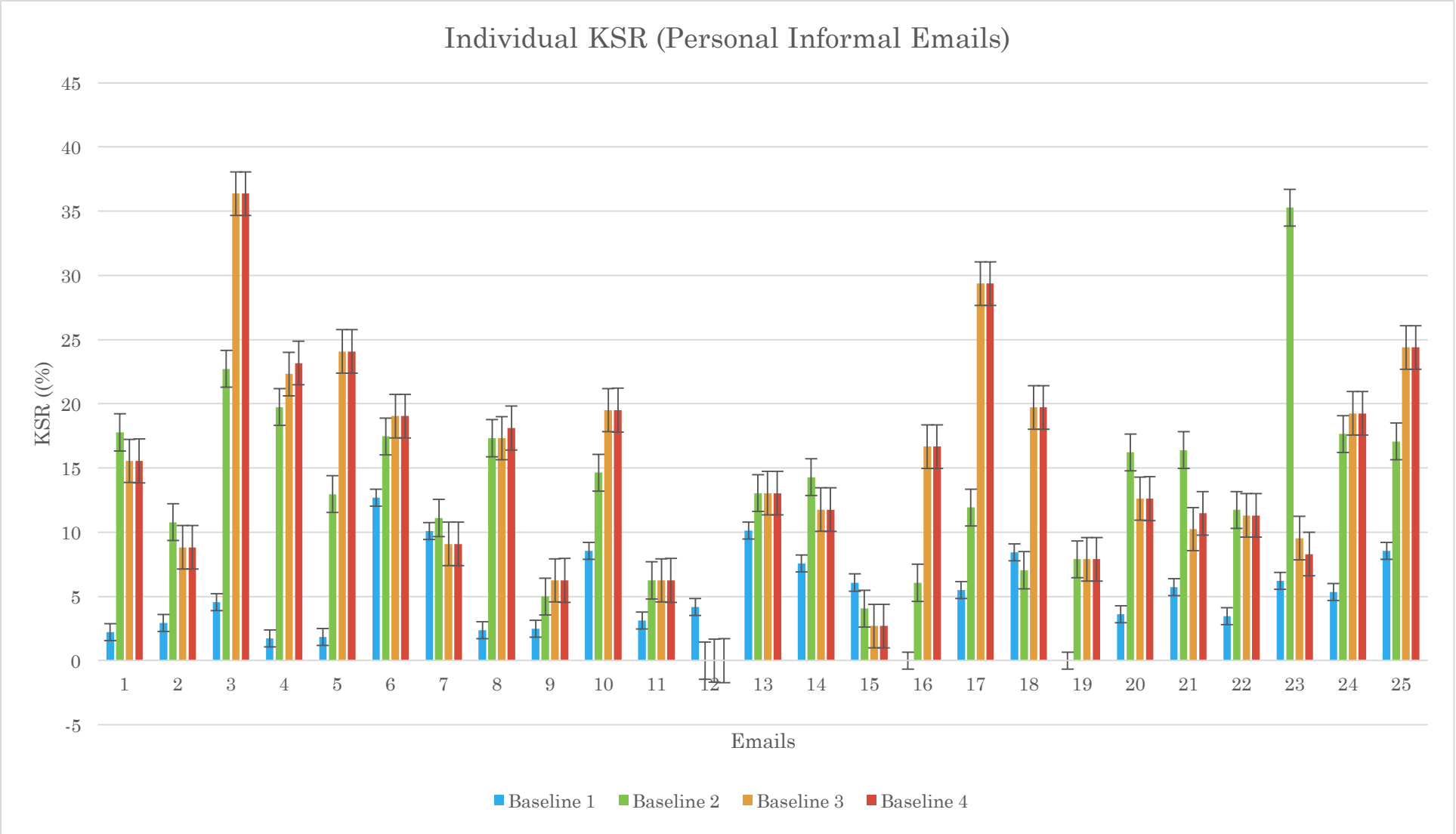


*Figure 6.10 Individual KSR for Personal Informal Emails*

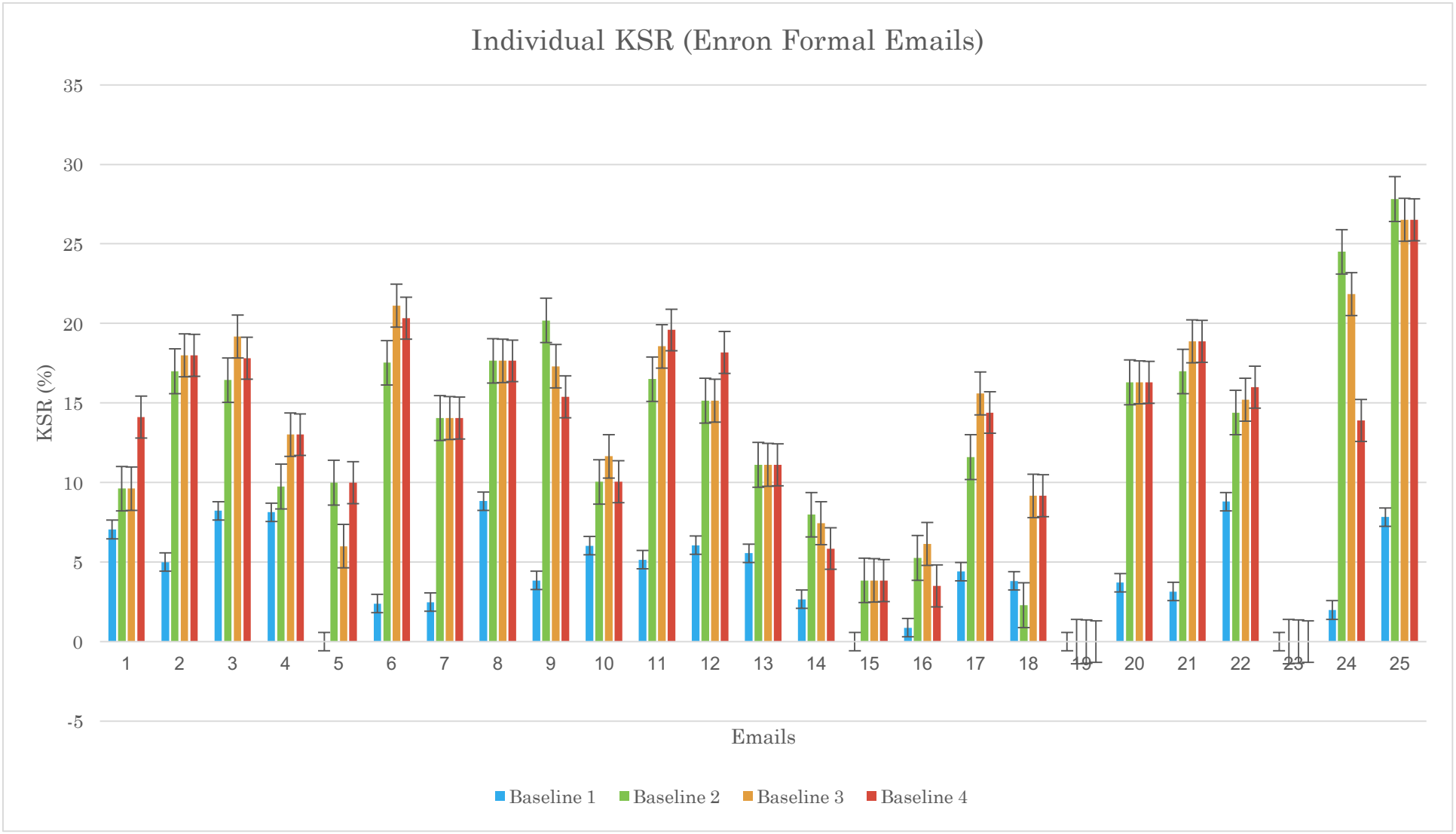


*Figure 6.11 Individual KSR for Enron Formal Email*

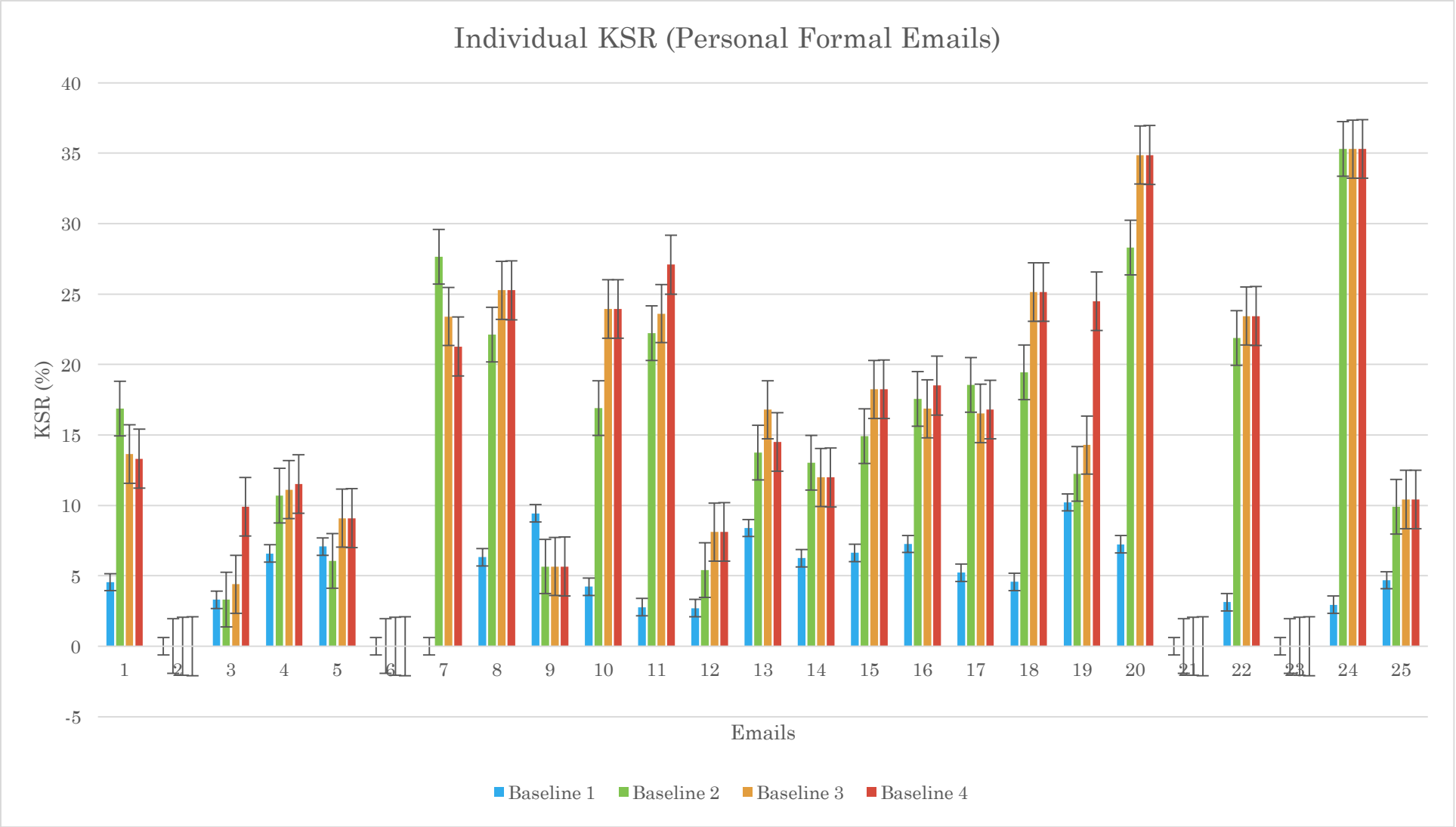


*Figure 6.12 Individual KSR for Personal Formal Emails*

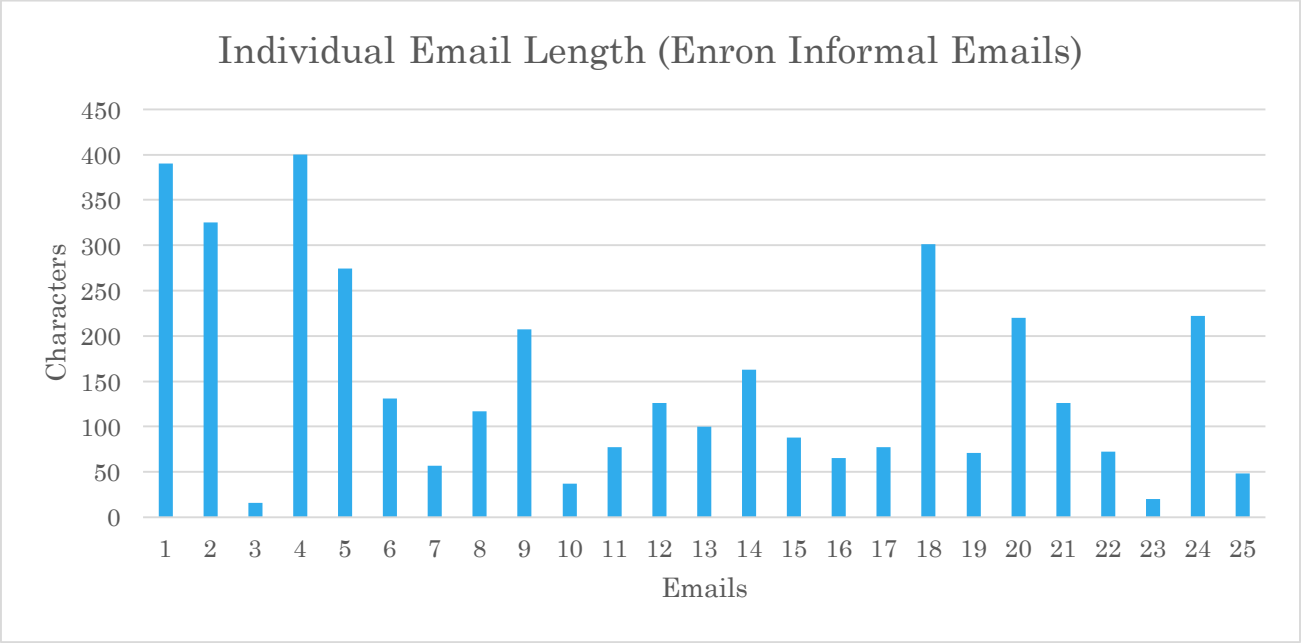


*Figure 6.13 Individual Email Length for Enron Informal Emails*

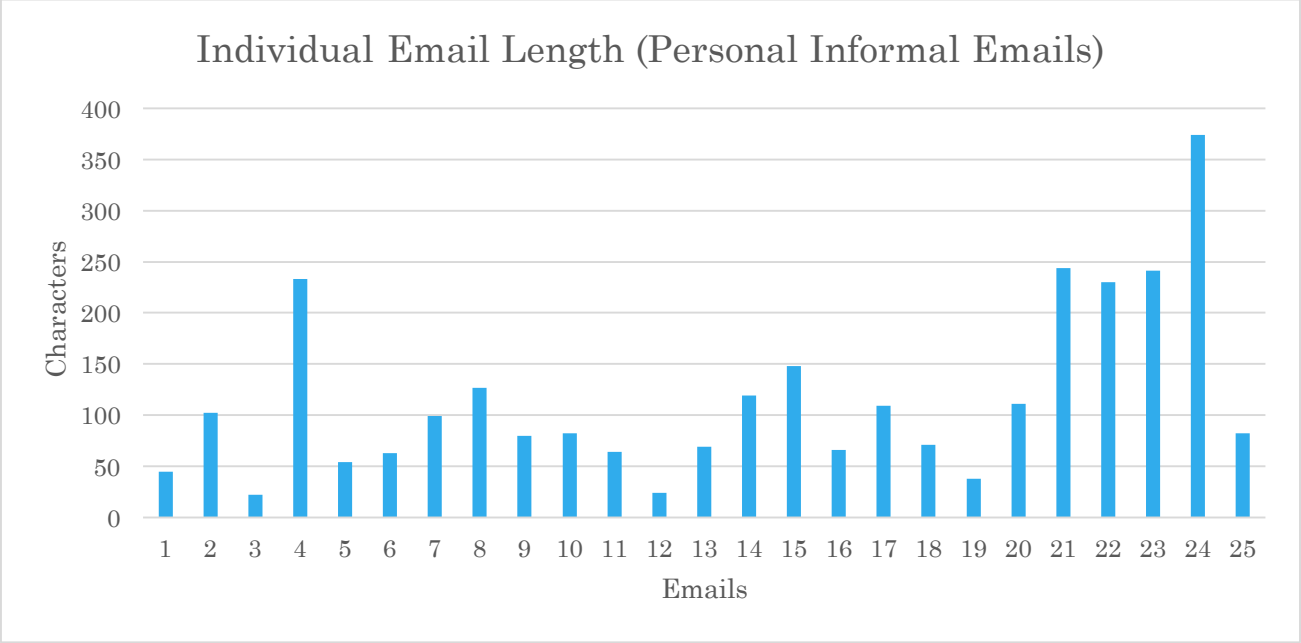


*Figure 6.14 Individual Email Length for Personal Informal Emails*

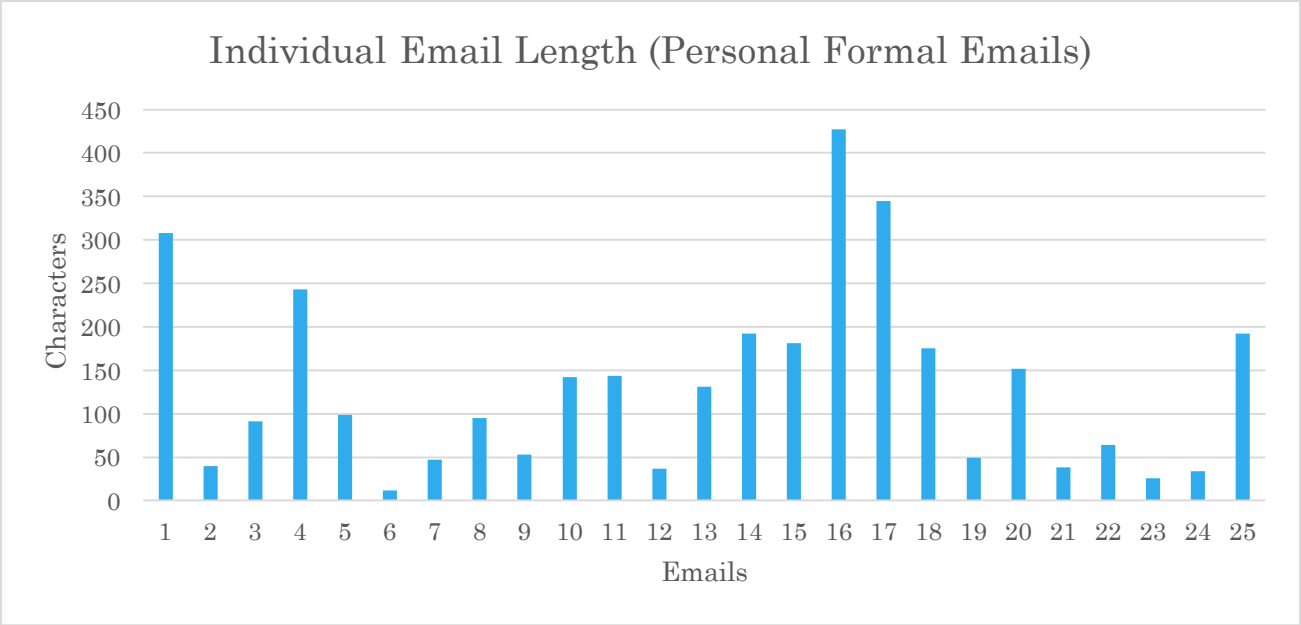


*Figure 6.15 Individual Email Length for Personal Formal Emails*

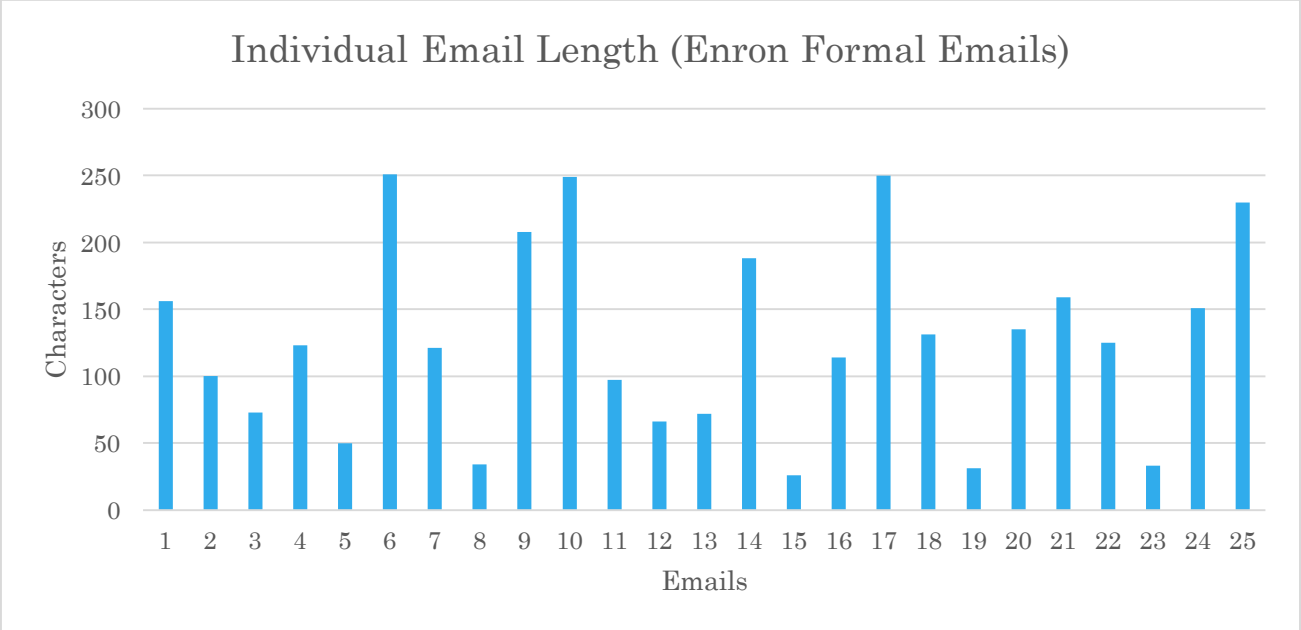


*Figure 6.16 Individual Email Length for Enron Formal Emails*

Additional statistics were calculated to identify which is the most frequently used corpus (or not found) from which the predictions are generated by using the MLE i.e.,

$$normalised\ counts = \frac{No.\ times\ prediction\ taken\ from\ corpus}{Total\ number\ of\ wo\square ds\ in\ test\ document} \tag{6.3}$$

From the Equation 6.3, the higher the normalised counts, the greater number of predictions generated from the particular corpus. Figure 6.17 illustrates the normalised counts across all emails for each training corpus. From the graph, it is evident that UBC has the highest number of predictions taken from it, and the BNC has the lowest. The significance of these results is discussed further in the Discussion and Future Works sections.

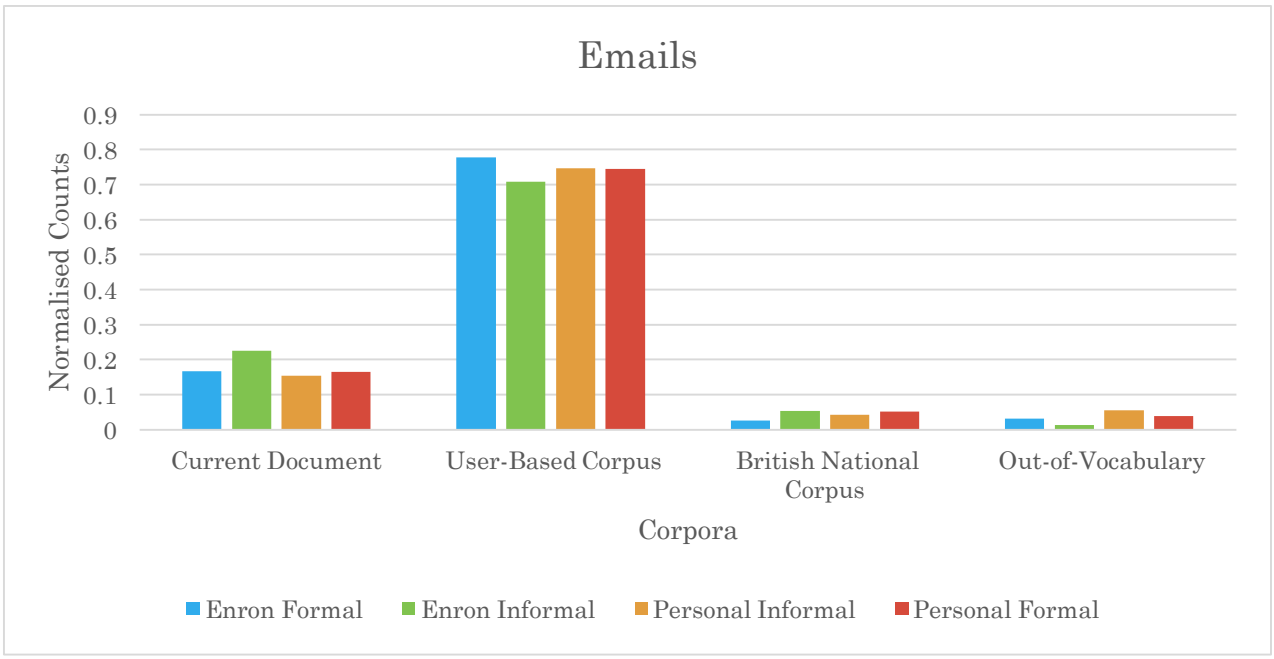


*Figure 6.17 Normalised counts for all corpora for each test set*

# 7 Discussion

## 7.1 Summary of Key Findings

- Baselines 2 – 4 all outperformed Baseline 0 and Baseline 1 with respect to KSR, but were statistically similar compared to each other
- Baselines 2-4 had reduced KT in comparison to Baseline 0 and Baseline 1, but were statistically similar compared to each other
- No apparent correlation between email length and KSR
- KSR was similar across all test sets with no major discrepancies between Formal vs. Informal or Enron vs. Personal emails

## 7.2 Aims

The aims of the tests were to identify the following:

1. Is the model capable of word prediction?
2. Does a context-driven UBC yield more accurate results?
3. Is there any added benefit of including WordNet predictions?

The following sections will address these aims with respect to the results obtained.

### *7.2.1 Aim 1*

The overall aim of this project was to build a user-based word prediction model, whose predictions are based largely on the user rather than the language. While the model itself was designed to reflect this, the tests were implemented to verify if the model was capable of word prediction on some unseen test data. From the results, it is evident that all baseline variations of the model, including the most basic unigram model (Baseline 1), is capable of word prediction.

Additionally, the improvement of Baseline 2 upon Baseline 1 for KT and KSR signifies the benefit of using bigram predictions than unigram predictions. This

is as expected, as unigram predictions do not take into consideration the input or context word when deriving the appropriate predictions, rather purely considering the frequency of words in a corpus.

### *7.2.2 Aim 2*

From the results (both KT and KSR), Baseline 2 and Baseline 3 were approximately the same with any small differences deemed as statistically insignificant. This suggests that having a task-driven UBC (Baseline 3) does not improve up on the accuracy of word prediction. Theoretically, tailoring the UBC to the context of what is being written should generate more accurate predictions particularly in this system, as the prediction list contains a portion of all possible predictions. For example, if the user is writing an informal email to his friend Bob and begins the email with 'Hi', the informal emails UBC is more likely to contain the bigram 'Hi Bob' more frequently than the generic UBC, which may contain a range of bigrams beginning with 'Hi' but not necessarily ending with 'Bob'. Hence, the prediction list from the informal emails UBC has a higher likelihood of containing 'Bob' than from the generic UBC, which could contain a range of other names.

However, a potential reason for the lack of impact of a context-derived UBC could be due to crossover in content between different contexts. For example, formal content i.e., this project, is generally used in the context of a formal email however, could still be referenced in an informal context. In this case, the informal emails UBC is unlikely to contain any information regarding the formal content and therefore the prediction list either contains irrelevant predictions taken from the BNC or no predictions at all resulting in a greater KT and smaller KSR.

Apart from increasing accuracy of the predictions, the purpose of a task-driven UBC was to reduce the time taken to generate predictions (smaller corpus). Given that Baseline 3 is not impeding the word prediction performance, future versions could focus on trying to improve its accuracy rather than not

considering context at all. For example, ensuring that the task-driven UBC is regularly updated such that it can capture any new crossover content. This is discussed further in the Future Works section.

### *7.2.3 Aim 3*

The inability of Baseline 4 to reduce KT or increase KSR in comparison to Baseline 3 indicates that the WordNet predictions did not provide any added benefit to the model. There could be several reasons for this, including the misidentification of the synset leading to inaccurate synonymous predictions.

The identification of a synset is based on the degree of overlap between the previously written words, and the synset definition from WordNet. It is clear that the more previously written words or more semantically-related words will yield in a more correct identified synset. In the case of the test emails, they are quite short and jump from topic-to-topic quickly, making it difficult for WSD to work properly leading to synset misidentification and irrelevant synonyms being added to the prediction list. As such, WordNet predictions may prove to be more useful for longer and topic-dependent documents i.e., a report on the brain.

Based on this, hypothetically longer emails in the test sets should have resulted in Baseline 4 potentially outperforming the other baselines as there is a higher likelihood of the correct synset being identified. However, the results obtained show no apparent correlation between email length and the KSR for Baseline 4. This could mean that WordNet synonyms may be more useful for topic-dependent documents.

Another potential reason for the poor performance of WordNet could be that the predictions are synonymous to the input word and therefore, may not be as accurate for predicting the output word. The reason for basing the prediction on the input word is because the input word is known, and therefore there is more control over the predictions generated.

### *7.2.4 Limitations*

It is important to recognise the presence of certain limitations which can impede on the model's performance. Identifying these limitations can aid in understanding how to either reduce or eliminate them in future versions.

From Figure 6.17, it is clear that the UBC generates the most predictions, and consequently has the largest impact on the resulting accuracy of the predictions. The issue of crossover content impacting the performance of Baseline 3 can therefore potentially be explained by the limitation of using a predesigned UBC. Using a predesigned UBC means that the UBC does not contain the most up-to-date user information increasing the chance of crossover content. However, while the problem of crossover content cannot be eliminated entirely; it can only be reduced by ensuring that the user-based predictions are taken from the most recent representation of the user.

Additionally, the final syntactic and semantic predictions listed were only a portion of *all* the possible predictions (3 each). While it is unrealistic to list all these predictions, the portion of predictions listed could affect the accuracy of the model. For example, the more predictions listed, the higher the likelihood of one of the predictions being correct. However, given that the number of predictions listed is dependent on the final UCI (i.e., how many predictions can fit into the screen at a time), this limitation may be unavoidable.

An evaluation limitation is the calculation of the keystrokes on the test data. The keystrokes were calculated purely on the characters of the email thereby disregarding the keystrokes involved in selection of the prediction i.e., choosing the predictions via corresponding numbers and the enter key for selection. The main reason for using this particular evaluation method is because the aims of the tests were to evaluate the accuracy of the predictions rather than the system as a whole.

### *7.2.5 Significance*

The tests were designed to identify if the language model based on the principles of SLM adaptation and back-off modelling is a feasible word prediction model, and evaluate the advantages and disadvantages of certain design implementations such as the inclusion of WordNet or using a task-driven UBC.

The results obtained for this 'proof-of-concept' confirm the model's potential as a word prediction model while highlighting possible modifications to further increase the accuracy of the predictions. Specifically, the results indicate that there is no statistically significant difference between using a task-driven or a generic UBC, and there is no significant benefit from the inclusion of WordNet on the accuracy of predictions.

At face value, the results indicate that there is no statistically significant difference between Baseline 2, 3 or 4 and therefore technically, any of the baselines can be used to improve upon the future word prediction model. However, there are other factors that should be considered such as time. While time as a metric has not been validated by the tests, intuitively Baseline 3 and Baseline 4 use a task-based (therefore smaller) UBC and consequently, may have a smaller time associated with generating predictions than Baseline 2, which uses a much larger UBC. As such, Baseline 3 may be a better choice for future improvements instead. However, to make any conclusive decisions, further tests particularly involving time as a metric must be evaluated first.

The results strongly suggest that WordNet offers no benefit to prediction model when writing an email. However, WordNet's influence for other tasks, particularly those requiring a longer and topically-similar document needs to be further tested. Given that the prediction model is to be used for a range of tasks and contexts, depending on the results obtained for other tasks, appropriate decisions can be made on WordNet's inclusion, or potentially partial inclusion using a weighting scheme.

# 8 Conclusions and Future Work

## 8.1 Future Work

There are two main aspects to the future work section – future work involving the word prediction model and generation of user-based predictions, and future work involving development of the final UCI.

#### *8.1.1.1 Word Prediction Model*

##### 8.1.1.1.1 Updating User-Based Corpus

The results highlighted the problem of crossover content, and arose primarily due to the pre-generated corpora used in the model. As mentioned previously, a way to address this is to continually update the contextual UBCs as frequently as possible (i.e., each night) to ensure that the UBC contains the most up-to-date information on the user. For this to be done efficiently, there must be a mechanism of storing the documents (emails, reports, webpages searched and saved etc.) at regular intervals. There could be several methods of implementation, i.e., interfacing with existing email systems such as Gmail to automatically download new emails and caching, as well as downloading, web content. If this is achieved, then theoretically the accuracy of the predictions should increase proportionally increasing the overall performance of the model.

##### 8.1.1.1.2 Static and Dynamic Weights

The current model works as back-off model and hence, each corpus has an equal weight. However, in reality, predictions are more likely to come from one type of corpus than another, for example, the likelihood of predictions being generated from the UBC might be much larger than predictions being generated from the BNC. This is in fact validated by preliminary statistics seen in Figure 6.17, indicating that the likelihood of predictions taken from UBC is more than double than that of the current document or BNC. Aside from the three main corpora,

the same trend is applicable for unfiltered (syntactic) vs. filtered (semantic) corpora, as well as WordNet.

To reflect these various influences, weights can be added for each corpora or source, of predictions to form a linearly interpolated model i.e.,

$$P\ (C) = \big(W_X(W_S(P\ (C|X_1)) * W_s(P(C|X_2))) + W_Y(W_S(P\ (C|Y_1)) * (W_sP(C|Y_2))) + W_Z(W_S(P\ (C|Z_1)) * W_s(P(C|Z_2)))\big) + W_N \quad (8.1)$$

where $W_X$, $W_Y$, $W_Z$ and $W_N$ are the associated weights of each corpus and WordNet, and $W_S$ and $W_s$ are the weights for the unfiltered (syntactic predictions) and filtered (semantic predictions) respectively. The three corpora (current document, UBC and BNC) are given by Corpus X, Corpus Y and Corpus Z respectively, with $X_1$, $Y_1, Z_1$signifying the unfiltered versions and $X_2$, $Y_2$ and $Z_2$ signifying the filtered versions.

Due to time constraints, preliminary weight were only calculated for each of the corpora $W_X$, $W_Y$ and $W_Z$ (Table 8.1). The calculation of MLE for these weights are shown in Equations (8.2-8.4) and results tabulated (Table 8.2). Given that there is a small portion of words that were not found in the corpus (i.e., OOV) and to ensure that the weights all add up to 1, the normalised count for the OOV words was distributed evenly across the three corpora weights. Hence, the final raw weights are the average of all the weights for each test set plus the evenly distributed average OOV normalised count.

It is important to realise that these weights are *preliminary* and therefore require further tuning in the future. A way of achieving this fine tuning would be to simply run a greater amount of unseen test data through the model.

$$W_X = \frac{No.\,times\ prediction\ taken\ from\ BNC}{Total\ number\ of\ words\ in\ test\ document} \quad (8.2)$$

$$W_Y = \frac{No.times\ prediction\ taken\ from\ UBC}{Total\ number\ of\ wor\square s\ in\ test\ document} \quad (8.3)$$

$$W_Z = \frac{No.times\ prediction\ taken\ from\ current\ document\ corpus}{Total\ number\ of\ words\ in\ test\ document} \quad (8.4)$$

*Table 8.1 Normalised Counts for each corpora across all test sets*

| **Normalised Counts** | **Current Document ($W_Z$)** | **UBC ($W_Y$)** | **BNC ($W_X$)** | **OOV** |
|---|---|---|---|---|
| **Enron Formal Emails** | 0.166 | 0.777 | 0.026 | 0.031 |
| **Enron Informal Emails** | 0.226 | 0.709 | 0.053 | 0.013 |
| **Personal Informal Emails** | 0.155 | 0.747 | 0.042 | 0.056 |
| **Personal Formal Emails** | 0.165 | 0.745 | 0.052 | 0.038 |
| **Average** | 0.178 | 0.744 | 0.043 | 0.035 |

*Table 8.2 Preliminary Weights for Linear Interpolation Model*

| **Weights** | **Weight Values** |
|---|---|
| $\boldsymbol{W_X}$ | 0.189 |
| $\boldsymbol{W_Y}$ | 0.756 |
| $\boldsymbol{W_Z}$ | 0.055 |

These corpora weights are *static* in that they remain the same throughout the program. However, in reality, as the document progresses, these weights may change. For example, as the user's current document becomes longer, the

likelihood of the predictions taken from the current document may surpass the likelihood of the other corpora. Weights that are able to adapt to these changes are *dynamic* weights. Calculating dynamic weights can be done in many ways, including simply doing a real-time calculation of where the predictions are being taken from and adjusting the weights within a particular time or 'word'-frame i.e., adjust the weights after 100 words.

#### 8.1.1.1.3 Recency Model

A recency model can be incorporated to weight information inside the corpus itself, particularly the UBC. For example, predictions based on emails or webpages written or accessed in the last week as opposed to the last year may be weighted higher as the user may be more likely to write similar to their recent emails/documents rather than former email/documents. In this case, time is the metric used to evaluate recent vs. old, however another possible metric to be used could be words. That is, rank the words in the corpus based on the *frequency* of usage rather than *when* they were used. For example, if the user wrote an essay on amphibians three years ago, but now wishes to write a similar report, predictions based on this three-year old report will be weighted higher under the word-based recency model as 'amphibian' and related terms would be ranked higher due to their frequency of use rather than a time-based recency model which would decrease the predictions' weight as it was accessed a long time back.

Recency language models are important for various disciplines including information retrieval, where the retrieved documents may need to be both topically and within a certain time frame relevant to the user's query. Li and Croft (2003) designed a time-based language model which incorporates a time-based posterior prior for each document in the language model, therefore when selecting the documents via a ranking process, the documents will be subjected to a time constraint and ranked accordingly. This method could potentially be translated for the word prediction model, where the 'documents' could be emails,

PDFs and so forth each with a time-based prior and therefore ranked within the corpus solely based on these priors.

#### 8.1.1.1.4 Syntactic Information

Incorporating syntactic information in a language model (see Literature Review) is useful in eliminating inappropriate or inaccurate predictions. For example, if the user wants the write the sentence 'my aunt is going to the shops', the prediction after 'aunt' must be *is* and not *are* as the latter defies the subject + verb agreement rule. This is particularly important for this model, as the predictions are largely based on the user and as such, if the user has poor grammar in their emails or documents, this will follow through as they are typing new documents.

Additionally, in English, pronouns are often used in replacement of repeated nouns. For example, instead of writing 'Jack climbed over the fence and then Jack fell down', we write 'Jack climbed over the fence and then he fell down'. That is, *he* replaces *Jack* (a proper noun) in the sentence. To reflect this writing style, predicting pronouns instead of repeating nouns can be incorporated into the language model as part of syntactic information. The implementation of this can be quite complex, however a simplistic initial method could be store all real-time proper nouns in a list using a part-of-speech tagger. If the proper noun is used more than once, include the pronoun as a prediction each time the proper noun is suggested. The crux of the problem, however, is how to identify the correct pronoun for the subject i.e., how does the model know that 'him' is the correct pronoun for 'Jack'? This must be investigated further in future versions.

#### 8.1.1.1.5 Perplexity Evaluation

The primary evaluation metrics used for the results were KT and KSR, however an equally important evaluation metric is perplexity (see Background section). As mentioned, perplexity is a measure of how much probability the trained

language model assigns to each word on some unseen test data. The more information provided by the language model, the lower the perplexity.

To measure perplexity, the language model must be first trained by assigning *probabilities* (not counts as what was used for generating predictions) to each word for both the UBC and the BNC. For a bigram model, the probability is given by Equation 2.4. Once the model has been trained, it can be used to assign probabilities on some unseen test-data. Perplexity is extremely sensitive to zero probabilities (i.e., unknown words) and therefore must use some form of smoothing to eliminate these zero probabilities.

Training the model takes significant computational power and time given the number of words in UBC (1.2 million) and BNC (100 million). Consequently, given the time constraints and computational limitations when undertaking this project, the results were not able to be generated.

#### *8.1.1.2 Unconscious Computer Interface*

The UCI is part of the larger project where the predictions from the language model are presented in an alphabetized list to the user (Figure 8.1). Using BCI technology, the user can select the appropriate predictions and append it to the current document.

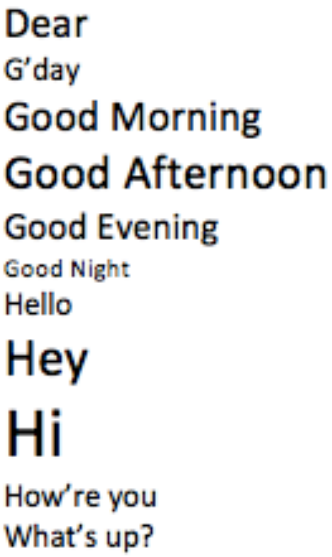


*Figure 8.1 UCI Selection Example*

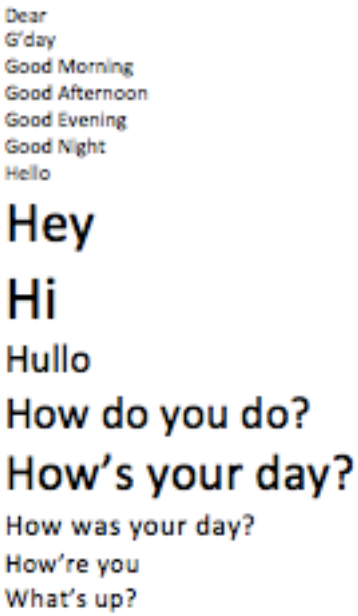


*Figure 8.2 UCI Selection Example*

#### 8.1.1.2.1 Selection

The selection of the UCI can be explained using the examples shown in Figures 8.1 and 8.2. Figure 8.1 depicts the initial prediction list shown to the user. Each

prediction has an associated probability and its size is proportional to that probability i.e., the most probable words in the initial prediction list are 'Good Afternoon', 'Hey' and 'Hi'. The user can peruse through the list using BCI technology and as the user focuses on a particular prediction or section of the prediction list, the associated section enlarges with new predictions emerging (Figure 8.2). In the example, the user focuses on the word 'Hey', and predictions on either side of the word are essentially 'greeked out' while the predictions under the 'H' section are enlarged. Once the user selects the appropriate prediction using BCI technology, the prediction is appended to the final document and a new prediction list based on the newly appended prediction is displayed.

For simplicity, the examples shown in Figures 8.1 and 8.2 are in black and white, however, more complicated versions can colour code the predictions according to the source and style. For example, the three corporal sources of predictions in the model are the current document, UBC and BNC, and along with WordNet, each can have their own associated colour. Similarly, syntactic and semantic predictions can each have their own colour.

#### 8.1.1.2.2 Evaluation

The predictions generated from the language model must be adapted to fit their use in the UCI. Essentially, the predictions in the UCI should resemble a probability search tree with the branches as predictions and their associated probabilities. As the user focuses their attention on a particular word, new predictions with new probabilities are added as branches until the correct word (information) is found. Hence, a potential evaluation metric for the UCI could be to count the number of steps, or branches in the search tree from the initial highest probable word until the correct word. In the final UCI, the 'steps' are measured by BCI however, a more 'proof-of-concept' approach would be to simply use the direction (up, down, left, right) keys where the 'up' and 'down' keys could be used to peruse the list, 'left' to make a selection and 'right' as a form of delete.

Hence, the number of steps would simply be the number of times the 'left' key was pressed.

## a. Summary

The designed model is capable of providing predictions based largely on the user rather than solely the language, with the predictions reflecting the user's interests, style of writing and their current task being undertaken. Results show that this model is capable of word prediction, whilst also highlighting potential design modifications such as use of a generalised corpus opposed to a context-derived corpus and further evaluating the effect of WordNet. Aside from the significance of developing a feasible user-based word prediction language model, the significance of the results also means that this tool can be successfully integrated with the future UCI.

# References

Akram, F., Metwally, M.K., Han, H.S., Jeon, H.J. and Kim, T.S., 2013, February. A novel P300-based BCI system for words typing. In *Brain-Computer Interface (BCI), 2013 International Winter Workshop on* (pp. 24-25). IEEE.

Akram, F., Han, S. M. & Kim, T. S. 2015. An efficient word typing P300-BCI system using a modified T9 interface and random forest classifier. *Comput Biol Med,* 56**,** 30-6.

A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39:1–38, 1977.

Baeza-Yates, R. and Ribeiro-Neto, B., 1999. *Modern information retrieval* (Vol. 463). New York: ACM press.

Banerjee, S. and Pedersen, T., 2002, February. An adapted Lesk algorithm for word sense disambiguation using WordNet. In *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 136-145). Springer Berlin Heidelberg.

Bellegarda, J. R. 2000. Exploiting latent semantic information in statistical language modeling. *Proceedings of the IEEE,* 88**,** 1279-1296.

Bellegarda, J.R., 2004. Statistical language model adaptation: review and perspectives. *Speech communication*, *42*(1), pp.93-108.

Berger, A. L., Pietra, V. J. D. & Pietra, S. A. D. 1996. A maximum entropy approach to natural language processing. *Computational linguistics,* 22**,** 39-71.

Blei, D. M., Ng, A. Y. & Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of machine Learning research,* 3**,** 993-1022.

Boyd-Graber, J. L. & Blei, D. M. Syntactic topic models. Advances in neural information processing systems, 2009. 185-192.

British National Corpus. 2009. *What is the BNC?* [ONLINE] Available at: http://www.natcorp.ox.ac.uk/corpus/. [Accessed 4 October 2016].

Brodwin MG, Siu FW, Howard J, Brodwin ER 2009, *Medical, Psychosocial and Vocational Aspects of Disability*, Elliott & Fitzpatrick Inc. GA, viewed 7th June 2016,
<http://www.kvccdocs.com/KVCC/2014Spring/MHT226/Medical%20Aspects.pdf#page=299)>

Charniak, E. 2000. A maximum-entropy-inspired parser. *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference.* Seattle, Washington: Association for Computational Linguistics.

Charniak, E. 2001. Immediate-head parsing for language models. *Proceedings of the 39th Annual Meeting on Association for Computational Linguistics.* Toulouse, France: Association for Computational Linguistics.

Chelba, C. and Jelinek, F., 1998, August. Exploiting syntactic structure for language modeling. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1* (pp. 225-231). Association for Computational Linguistics.

Chelba, C. & Jelinek, F. 2000. Structured language modeling. *Computer Speech & Language,* 14**,** 283-332.

Coccaro, N. & Jurafsky, D. Towards better integration of semantic predictors in statistical language modeling. ICSLP, 1998. Citeseer.

Collins, M., Roark, B. & Saraclar, M. 2005. Discriminative syntactic language modeling for speech recognition. *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. Ann Arbor, Michigan: Association for Computational Linguistics.

Dempster, A. P., Laird, N. M. & Rubin, D. B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)***,** 1-38.

Don Johnston. 2016. *Co: Writer Universal*. [ONLINE] Available at: http://donjohnston.com/cowriter/. [Accessed 30 June 2016].

Enron Email Dataset. 2015. *Enron Email Dataset*. [ONLINE] Available at: https://www.cs.cmu.edu/~./enron/. [Accessed 4 October 2016].

Erdogan, H., Sarikaya, R., Gao, Y. & Picheny, M. Semantic structured language models. INTERSPEECH, 2002.

Farwell, L. A. & Donchin, E. 1988. Talking off the top of your head: toward a mental prosthesis utilizing event-related brain potentials. *Electroencephalography and Clinical Neurophysiology,* 70**,** 510-523.

Fraser Shein, Vivian Tsang. 2001. *WordQ Writing Software*. [ONLINE] Available at: http://www.goqsoftware.com/pdf/research/wordQ4writingsoftware.pdf. [Accessed 30 June 2016].

Gildea, D. & Hofmann, T. 1999. Topic-based language models using EM. *History,* 11111**,** 11111.

Good, I.J., 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, *40*(3-4), pp.237-264.

Goodman, J., Venolia, G., Steury, K. & Parker, C. Language modeling for soft keyboards. Proceedings of the 7th international conference on Intelligent user interfaces, 2002. ACM, 194-195.

Griffiths, T. L., Steyvers, M., Blei, D. M. & Tenenbaum, J. B. Integrating topics and syntax. Advances in neural information processing systems, 2004. 537-544.

Guerreiro, T.J.V. 2007, Assistive Technologies for Spinal Cord Injured Individuals A Survey

Huang, J.H. and Powers, D., 2001. Large scale experiments on correction of confused words. *Australian Computer Science Communications*, *23*(1), pp.77-82.

Katz, S. 1987. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE transactions on acoustics, speech, and signal processing,* 35**,** 400-401.

Kaufmann, T. & Pfister, B. 2012. Syntactic language modeling with formal grammars. *Speech Communication,* 54**,** 715-731.

Khudanpur, S. & Wu, J. 2000. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech & Language,* 14**,** 355-372.

Klakow, D. Log-linear interpolation of language models. ICSLP, 1998.

Kneser, R. & Steinbiss, V. On the dynamic adaptation of stochastic language models. Acoustics, Speech, and Signal Processing, 1993. ICASSP-93., 1993 IEEE International Conference on, 1993. IEEE, 586-589.

Kuhn, R. & De Mori, R. 1990. A cache-based natural language model for speech recognition. *IEEE transactions on pattern analysis and machine intelligence,* 12**,** 570-583.

Landauer, T. K. & Dumais, S. T. 1997. A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review,* 104**,** 211.

Laureys, S., Pellas, F., Van Eeckhout, P., Ghorbel, S., Schnakers, C., Perrin, F., Berre, J., Faymonville, M. E., Pantke, K. H., Damas, F., Lamy, M., Moonen, G. & Goldman, S. 2005. The locked-in syndrome : what is it like to be conscious but paralyzed and voiceless? *Prog Brain Res,* 150**,** 495-511.

Li, X. and Croft, W.B., 2003, November. Time-based language models. In *Proceedings of the twelfth international conference on Information and knowledge management* (pp. 469-475). ACM.

Lesk, M., 1986, June. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation* (pp. 24-26). ACM.

Lesher, G. W. & Rinkus, G. J. Domain-specific word prediction for augmentative communication. Proceedings of the RESNA 2002 Annual Conference, 2002.

Li, J. & Hirst, G. Semantic knowledge in word completion. Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility, 2005. ACM, 121-128.

Mackenzie, I. S. & Zhang, X. Eye typing using word and letter prediction and a fixation algorithm. Proceedings of the 2008 symposium on Eye tracking research & applications, 2008. ACM, 55-58.

Mahajan, M., Beeferman, D. & Huang, X. Improved topic-dependent language modeling using information retrieval techniques. Acoustics, Speech, and Signal Processing, 1999. Proceedings., 1999 IEEE International Conference on, 1999. IEEE, 541-544.

Manning, C.D. and Schütze, H., 1999. *Foundations of statistical natural language processing* (Vol. 999). Cambridge: MIT press.

Manyakov, N. V., Chumerin, N., Combaz, A. & VAN HULLE, M. M. 2011. Comparison of classification methods for P300 brain-computer interface on disabled subjects. *Comput Intell Neurosci,* 2011**,** 519868.

Jurafsky, D and Martin, J.H., 2000. Speech and language processing. *International Edition*, *710*.

Mitchell, J and Lapata, M. Language models based on semantic composition. Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1, 2009. Association for Computational Linguistics, 430-439.

Mora-Cortes, A., Manyakov, N. V., Chumerin, N. & Van Hulle, M. M. 2014. Language model applications to spelling with brain-computer interfaces. *Sensors,* 14**,** 5967-5993.

Natural language Toolkit — NLTK 3.0 documentation 2015. <http://www.nltk.org> (Accessed: 4 October 2016).

Ney, H., Essen, U. and Kneser, R., 1994. On structuring probabilistic dependences in stochastic language modelling. *Computer Speech & Language*, *8*(1), pp.1-38.

Orhan, U., Hild, K. E., Erdogmus, D., Roark, B., Oken, B. & Fried-Oken, M.

RSVP keyboard: an EEG based typing interface. 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012. IEEE, 645-648.

Powers, D.M., 1997, July. Learning and Application of Differential Grammars. In *CoNLL* (pp. 88-96).

Princeton University "About WordNet." WordNet. Princeton University. 2010. <http://wordnet.princeton.edu>

Riccardi, G., Potamianos, A. & Narayanan, S. Language model adaptation for spoken language systems. ICSLP, 1998. 2327-2330.

Roark, B. 2001. Probabilistic top-down parsing and language modeling. *Computational linguistics,* 27**,** 249-276.

Rosenfeld, R., 1996. A maximum entropy approach to adaptive statistical language modeling.

Rosenfeld, R. 2005. *Adaptive statistical language modeling: A maximum entropy approach*. Department of the Navy, Naval Research Laboratory.

Ryan, D. B., Frye, G., Townsend, G., Berry, D., Mesa-G, S., Gates, N. A. & Sellers, E. W. 2010. Predictive spelling with a P300-based brain–computer interface: increasing the rate of communication. *Intl. Journal of Human–Computer Interaction,* 27**,** 69-84.

Schalk, G., Mcfarland, D. J., Hinterberger, T., Birbaumer, N. & Wolpaw, J. R. 2004. BCI2000: a general-purpose brain-computer interface (BCI) system. *IEEE Trans Biomed Eng,* 51**,** 1034-43.

Shepherd Centre 2011, *Understanding Spinal Cord Injury,* viewed 7th June 2016, < http://www.spinalinjury101.org >

Speier, W., Arnold, C., Lu, J., Taira, R. K. & Pouratian, N. 2011. Natural language processing with dynamic classification improves P300 speller accuracy and bit rate. *Journal of neural engineering,* 9**,** 016004.

Spinal Hub 2016, *Spinal Nerves Up Close*. [ONLINE] viewed 5th October 2016, < http://www.spinalhub.com.au/what-is-a-spinal-cord-injury/what-happens-to-the-spinal-cord-after-injury/spinal-nerves-up-close>

Trnka, K. and McCoy, K.F., 2008, June. Evaluating word prediction: framing keystroke savings. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers* (pp. 261-264). Association for Computational Linguistics.

Trnka, K., Yarrington, D., Mccoy, K. & Pennington, C. 2005. Topic Modeling in Word Prediction For Aac. Technical Report.

Wallach, H. M. Topic modeling: beyond bag-of-words. Proceedings of the 23rd international conference on Machine learning, 2006. ACM, 977-984.

Wandmacher, T. & Antoine, J.-Y. 2008. Methods to integrate a language model with semantic information for a word prediction component. *arXiv preprint arXiv:0801.4716*.

Wills, S. A. & Mackay, D. J. 2006. DASHER-an efficient writing system for brain-computer interfaces? *IEEE Transactions on Neural Systems and Rehabilitation Engineering,* 14**,** 244-246.

Yang, D. and Powers, D.M., 2005, January. Measuring semantic similarity in the taxonomy of WordNet. In *Proceedings of the Twenty-eighth Australasian*

*conference on Computer Science-Volume 38* (pp. 315-322). Australian Computer Society, Inc..

Yang, D. and Powers, D.M., 2006. *Verb similarity on the taxonomy of WordNet*. Masaryk University.

# Appendix A – Clean BNC Corpus Code

```
import re
import io
import nltk
import os


os.chdir('/Users/madhumuralidharan/nltk_data/corpora/BNC/K/KS"') #choose BNC folder

def cleanhtml(raw_html): #filter html tags
  cleanr =re.compile('<.*?>')
  cleantext = re.sub(cleanr,'', raw_html)
  return cleantext

for i in os.listdir(os.getcwd()):
    if i.endswith(".txt"):
        print (i) #print the text file
        my_file = io.open(i,"r", encoding = "utf-8") #open text file
        text_file = my_file.read() #read
        new=cleanhtml(text_file) #clean
        new_file = open(i, "w") #rewrite old file with clean version
        new_file.write(new)
    else:
        continue
```

# Appendix B – Bigram Dictionary Code

```
def bigram(tokens):
    model=collections.defaultdict() #create new dictionary
    bigrams = nltk.bigrams(tokens)  #generate bigrams
    fdist=nltk.ConditionalFreqDist(bigrams) #generate conditional freq distribution from counts
    try:  #bigram frequency
        for k,v in fdist.items():
            model[k] = v #save bigram counts
    except KeyError:
        model[k]=1 #default if word is not found
    return model

model = bigram(tokens)
pickle.dump(model,open("bnc_probability.p","wb")) #save to file
```

# Appendix C – Model Implementation

```
def CheckUserInputExists(current_word, task):
    """
    :param current_word: current word for which predictions are generated
    :param task: email or document
    :return: prediction list
    The function takes the input word and the task (email/document). Depending on the task,
    the corresponding user-based corpus (informal/formal/document)  is selected. The system then looks for potential
    predictions first in the current document, then the user-based corpus and finally the BNC. Once the appropriate
    corpus is found, a conditional frequency distribution is generated containing the bigram counts from the corpus and
    used to generate the predictions.
    Note: Each of the corpora have a second filtered version (removed stop words and lemmatisation).
    If bigram predictions exist in any of two versions of the three corpora, the three most common/frequent
    predictions are stored in a list and returned.
    """
    new= []
    new = list(set(new))
    input = current_word
    overall_pred = []
    temp_corpus = task
    print(temp_corpus)

    #find appropriate user-based corpus

    if temp_corpus == 'formal':
        with io.open("Formal_Emails_Filtered_All.txt", "r", encoding="utf-8") as my_file:
            user_based_corpus = my_file.read()
    elif temp_corpus == 'informal':
        with io.open("Informal_Emails_Filtered_All.txt", "r", encoding="utf-8") as my_file:
            user_based_corpus = my_file.read()
    elif temp_corpus == 'document':
        with io.open("Past_PDFs.txt", "r", encoding="utf-8") as my_file:
            user_based_corpus = my_file.read()

    #Check for bigram predictions in current document
    if input in current_document:
        print("Found in current document")
        mynewtext = [w for w in current_document if w not in stopwords] #stopping
        new_text = (lemmatiser.lemmatize(i) for i in mynewtext) #lemmatising
        bigrams_normal = bigram_model(current_document,2)
        bigrams_semantic = bigram_model(new_text,2)
        cfd_normal = nltk.ConditionalFreqDist(bigrams_normal)
        cfd_semantic = nltk.ConditionalFreqDist(bigrams_semantic)
        normal_predictions = list(cfd_normal[input].most_common(3))
        semantic_predictions = list(cfd_semantic[input].most_common(3))
        if (normal_predictions == semantic_predictions): #if normal and semantic predictions are the same, store the first three in a list
```

```
            np = [] #temp prediction list
            normal_pred = []
            for i in range(len(normal_predictions)):
                np += normal_predictions[i]
                normal_pred = np[0::2]
            overall_pred = list(set((normal_pred)))
        else: #if normal and semantic predictions are not the same, build a new list with both the
normal and semantic predictions
            normal_pred = []
            semantic_pred = []
            np = []
            sp = []
            for i in range(len(normal_predictions)):
                np += normal_predictions[i]
                normal_pred = np[0::2]
            for j in range(len(semantic_predictions)):
                sp += semantic_predictions[j]
                semantic_pred = sp[0::2]
            overall_pred = (list(set(normal_pred) | set(semantic_pred))) #filter out common semantic
and syntactic predictions
        return overall_pred

    #check for bigram predictions in user-based corpus
    elif input in user_based_corpus:
        print("Found in user-based corpus")
        print("Found in BNC")
            cfd_normal = pickle.load(open("saved_ubc_cfd.p", "rb"))
            cfd_semantic = pickle.load(open("saved_ubc_semantic_cfd.p","rb"))
        normal_predictions = list(cfd_normal[input].most_common(3))
        semantic_predictions = list(cfd_semantic[input].most_common(3))
        if (normal_predictions == semantic_predictions):
            np = []
            normal_pred = []
            for i in range(len(normal_predictions)):
                np += normal_predictions[i]
                normal_pred = np[0::2]
                overall_pred = list(set((normal_pred)))
        else:
            normal_pred = []
            topic_pred = []
            np = []
            tp = []
            for i in range(len(normal_predictions)):
                np += normal_predictions[i]
                normal_pred = np[0::2]
            for j in range(len(semantic_predictions)):
                tp += semantic_predictions[j]
                topic_pred = tp[0::2]
            overall_pred = (list(set(normal_pred) | set(topic_pred)))
        return overall_pred
    #check for predictions in BNC
    elif input in corpus:
        print("Found in BNC")
            cfd_normal = pickle.load(open("saved_bnc_cfd.p", "rb"))
            cfd_semantic = pickle.load(open("saved_bnc_semantic_cfd.p", "rb"))
        normal_predictions = list(cfd_normal[input].most_common(3))
        semantic_predictions = list(cfd_semantic[input].most_common(3))
        if (normal_predictions == semantic_predictions):
```

```
        np = []
        normal_pred = []
        for i in range(len(normal_predictions)):
            np += normal_predictions[i]
            normal_pred = np[0::2]
        overall_pred = list(set((normal_pred)))
    else:
        normal_pred = []
        topic_pred = []
        np = []
        tp = []
        for i in range(len(normal_predictions)):
            np += normal_predictions[i]
            normal_pred = np[0::2]
        for j in range(len(semantic_predictions)):
            tp += semantic_predictions[j]
            topic_pred = tp[0::2]
        overall_pred = (list(set(normal_pred) | set(topic_pred)))
    return overall_pred
else:
    print("not found anywhere")
    overall_pred = []
    return overall_pred
```

# Appendix D – Individual Email KSR Values for Test Sets

| Personal Informal Emails | Baseline 0 (%) | Baseline 1 (%) | Baseline 2 (%) | Baseline 3 (%) | Baseline 4 (%) |
|---|---|---|---|---|---|
| **1** | 0 | 2.222±0.662 | 17.788±1.436 | 15.566±1.689 | 15.556±1.700 |
| **2** | 0 | 2.941±0.662 | 10.784±1.436 | 8.824±1.689 | 8.824±1.700 |
| **3** | 0 | 4.545±0.662 | 22.728±1.436 | 36.364±1.689 | 36.364±1.700 |
| **4** | 0 | 1.717±0.662 | 19.742±1.436 | 22.318±1.689 | 23.176±1.700 |
| **5** | 0 | 1.852±0.662 | 12.963±1.436 | 24.074±1.689 | 24.074±1.700 |
| **6** | 0 | 12.698±0.662 | 17.460±1.436 | 19.048±1.689 | 19.048±1.700 |
| **7** | 0 | 10.101±0.662 | 11.111±1.436 | 9.091±1.689 | 9.091±1.700 |
| **8** | 0 | 2.362±0.662 | 17.323±1.436 | 17.323±1.689 | 18.110±1.700 |
| **9** | 0 | 2.5±0.662 | 5±1.436 | 6.25±1.689 | 6.25±1.700 |
| **10** | 0 | 8.537±0.662 | 14.634±1.436 | 19.512±1.689 | 19.512±1.700 |
| **11** | 0 | 3.125±0.662 | 6.25±1.436 | 6.25±1.689 | 6.25±1.700 |
| **12** | 0 | 4.177±0.662 | 0±1.436 | 0±1.689 | 0±1.700 |
| **13** | 0 | 10.145±0.662 | 13.043±1.436 | 13.043±1.689 | 13.043±1.700 |
| **14** | 0 | 7.563±0.662 | 14.286±1.436 | 11.765±1.689 | 11.765±1.700 |
| **15** | 0 | 6.081±0.662 | 4.054±1.436 | 2.703±1.689 | 2.703±1.700 |
| **16** | 0 | 0±0.662 | 6.061±1.436 | 16.667±1.689 | 16.667±1.700 |
| **17** | 0 | 5.505±0.662 | 11.927±1.436 | 29.358±1.689 | 29.358±1.700 |
| **18** | 0 | 8.451±0.662 | 7.042±1.436 | 19.718±1.689 | 19.718±1.700 |
| **19** | 0 | 0±0.662 | 7.895±1.436 | 7.895±1.689 | 7.895±1.700 |

| | | | | | |
|---|---|---|---|---|---|
| **20** | 0 | 3.604±0.662 | 16.216±1.436 | 12.613±1.689 | 12.613±1.700 |
| **21** | 0 | 5.738±0.662 | 16.393±1.436 | 10.246±1.689 | 11.475±1.700 |
| **22** | 0 | 3.478±0.662 | 11.739±1.436 | 11.304±1.689 | 11.304±1.700 |
| **23** | 0 | 6.224±0.662 | 35.270±1.436 | 9.544±1.689 | 8.299±1.700 |
| **24** | 0 | 5.348±0.662 | 17.647±1.436 | 19.251±1.689 | 19.251±1.700 |
| **25** | 0 | 8.537±0.662 | 17.073±1.436 | 24.390±1.689 | 24.390±1.700 |

| Enron Informal Emails | Baseline 0 (%) | Baseline 1 (%) | Baseline 2 (%) | Baseline 3 (%) | Baseline 4 (%) |
|---|---|---|---|---|---|
| **1** | 0 | 7.949±0.484 | 13.077±1.261 | 13.333±1.307 | 14.872±1.311 |
| **2** | 0 | 6.462±0.484 | 14.462±1.261 | 11.077±1.307 | 12±1.311 |
| **3** | 0 | 0±0.484 | 31.25±1.261 | 31.25±1.307 | 31.25±1.311 |
| **4** | 0 | 2.75±0.484 | 13±1.261 | 14±1.307 | 14±1.311 |
| **5** | 0 | 5.109±0.484 | 16.423±1.261 | 16.423±1.307 | 16.423±1.311 |
| **6** | 0 | 4.580±0.484 | 16.031±1.261 | 19.084±1.307 | 22.137±1.311 |
| **7** | 0 | 7.018±0.484 | 15.789±1.261 | 7.018±1.307 | 7.018±1.311 |
| **8** | 0 | 5.128±0.484 | 13.675±1.261 | 15.385±1.307 | 15.385±1.311 |
| **9** | 0 | 2.415±0.484 | 18.841±1.261 | 22.705±1.307 | 22.705±1.311 |
| **10** | 0 | 5.405±0.484 | 8.108±1.261 | 10.811±1.307 | 10.811±1.311 |
| **11** | 0 | 9.091±0.484 | 18.182±1.261 | 19.481±1.307 | 19.481±1.311 |
| **12** | 0 | 1.587±0.484 | 7.143±1.261 | 26.190±1.307 | 26.190±1.311 |
| **13** | 0 | 1±0.484 | 21±1.261 | 16±1.307 | 16±1.311 |
| **14** | 0 | 2.454±0.484 | 4.908±1.261 | 12.270±1.307 | 12.270±1.311 |
| **15** | 0 | 7.954±0.484 | 10.227±1.261 | 7.955±1.307 | 7.955±1.311 |
| **16** | 0 | 1.538±0.484 | 6.154±1.261 | 20±1.307 | 20±1.311 |
| **17** | 0 | 5.195±0.484 | 6.493±1.261 | 7.792±1.307 | 7.792±1.311 |
| **18** | 0 | 6.312±0.484 | 14.286±1.261 | 16.279±1.307 | 15.947±1.311 |
| **19** | 0 | 1.408±0.484 | 14.085±1.261 | 19.718±1.307 | 19.718±1.311 |
| **20** | 0 | 5.455±0.484 | 13.182±1.261 | 15.455±1.307 | 15.455±1.311 |
| **21** | 0 | 7.143±0.484 | 13.492±1.261 | 19.048±1.307 | 18.254±1.311 |
| **22** | 0 | 4.177±0.484 | 19.444±1.261 | 18.056±1.307 | 18.056±1.311 |
| **23** | 0 | 5±0.484 | 0±1.261 | 0±1.307 | 0±1.311 |
| **24** | 0 | 5.405±0.484 | 19.369±1.261 | 20.721±1.307 | 20.720±1.311 |
| **25** | 0 | 4.177±0.484 | 12.5±1.261 | 12.5±1.307 | 18.75±1.311 |

| Personal Formal Emails | Baseline 0 (%) | Baseline 1 (%) | Baseline 2 (%) | Baseline 3 (%) | Baseline 4 (%) |
|---|---|---|---|---|---|
| **1** | 0 | 2.222±0.609 | 17.778±1.938 | 15.556±2.065 | 15.556±2.082 |
| **2** | 0 | 2.941±0.609 | 10.784±1.938 | 8.824±2.065 | 8.824±2.082 |
| **3** | 0 | 4.545±0.609 | 22.727±1.938 | 36.364±2.065 | 36.364±2.082 |
| **4** | 0 | 1.717±0.609 | 19.742±1.938 | 22.318±2.065 | 23.176±2.082 |
| **5** | 0 | 1.852±0.609 | 12.963±1.938 | 24.074±2.065 | 24.074±2.082 |
| **6** | 0 | 12.700±0.609 | 17.460±1.938 | 19.048±2.065 | 19.048±2.082 |
| **7** | 0 | 10.101±0.609 | 11.111±1.938 | 9.091±2.065 | 9.091±2.082 |
| **8** | 0 | 2.362±0.609 | 17.323±1.938 | 17.323±2.065 | 18.110±2.082 |
| **9** | 0 | 2.5±0.609 | 5±1.938 | 6.25±2.065 | 6.25±2.082 |
| **10** | 0 | 8.537±0.609 | 14.634±1.938 | 19.512±2.065 | 19.512±2.082 |
| **11** | 0 | 3.125±0.609 | 6.25±1.938 | 6.25±2.065 | 6.25±2.082 |
| **12** | 0 | 4.167±0.609 | 0±1.938 | 0±2.065 | 0±2.082 |
| **13** | 0 | 10.145±0.609 | 13.043±1.938 | 13.043±2.065 | 13.043±2.082 |
| **14** | 0 | 7.563±0.609 | 14.286±1.938 | 11.765±2.065 | 11.765±2.082 |
| **15** | 0 | 6.081±0.609 | 4.054±1.938 | 2.703±2.065 | 2.703±2.082 |
| **16** | 0 | 0±0.609 | 6.061±1.938 | 16.667±2.065 | 16.667±2.082 |
| **17** | 0 | 5.505±0.609 | 11.927±1.938 | 29.358±2.065 | 29.358±2.082 |
| **18** | 0 | 8.451±0.609 | 7.042±1.938 | 19.718±2.065 | 19.718±2.082 |
| **19** | 0 | 0±0.609 | 7.895±1.938 | 7.895±2.065 | 7.895±2.082 |
| **20** | 0 | 3.604±0.609 | 16.216±1.938 | 12.613±2.065 | 12.613±2.082 |
| **21** | 0 | 5.738±0.609 | 16.393±1.938 | 10.246±2.065 | 11.475±2.082 |
| **22** | 0 | 3.478±0.609 | 11.739±1.938 | 11.304±2.065 | 11.304±2.082 |
| **23** | 0 | 6.224±0.609 | 35.270±1.938 | 9.544±2.065 | 8.299±2.082 |
| **24** | 0 | 5.348±0.609 | 17.647±1.938 | 19.251±2.065 | 19.251±2.082 |
| **25** | 0 | 8.537±0.609 | 17.073±1.938 | 24.390±2.065 | 24.390±2.082 |

| Enron Formal Emails | Baseline 0 (%) | Baseline 1 (%) | Baseline 2 (%) | Baseline 3 (%) | Baseline 4 (%) |
|---|---|---|---|---|---|
| **1** | 0 | 7.051±0.581 | 9.615±1.405 | 9.615±1.355 | 14.103±1.317 |
| **2** | 0 | 5±0.581 | 17±1.405 | 18±1.355 | 18±1.317 |
| **3** | 0 | 8.219±0.581 | 16.438±1.405 | 19.178±1.355 | 17.808±1.317 |
| **4** | 0 | 8.130±0.581 | 9.756±1.405 | 13.008±1.355 | 13.008±1.317 |
| **5** | 0 | 0±0.581 | 10±1.405 | 6±1.355 | 10±1.317 |
| **6** | 0 | 2.390±0.581 | 17.530±1.405 | 21.116±1.355 | 20.319±1.317 |
| **7** | 0 | 2.479±0.581 | 14.050±1.405 | 14.050±1.355 | 14.050±1.317 |
| **8** | 0 | 8.824±0.581 | 17.647±1.405 | 17.647±1.355 | 17.647±1.317 |
| **9** | 0 | 3.846±0.581 | 20.192±1.405 | 17.308±1.355 | 15.385±1.317 |
| **10** | 0 | 6.024±0.581 | 10.040±1.405 | 11.647±1.355 | 10.040±1.317 |
| **11** | 0 | 5.155±0.581 | 16.495±1.405 | 18.557±1.355 | 19.588±1.317 |
| **12** | 0 | 6.061±0.581 | 15.152±1.405 | 15.152±1.355 | 18.182±1.317 |
| **13** | 0 | 5.556±0.581 | 11.111±1.405 | 11.111±1.355 | 11.111±1.317 |
| **14** | 0 | 2.660±0.581 | 7.979±1.405 | 7.447±1.355 | 5.851±1.317 |
| **15** | 0 | 0±0.581 | 3.846±1.405 | 3.846±1.355 | 3.846±1.317 |
| **16** | 0 | 0.877±0.581 | 5.263±1.405 | 6.140±1.355 | 3.509±1.317 |
| **17** | 0 | 4.4±0.581 | 11.6±1.405 | 15.6±1.355 | 14.4±1.317 |
| **18** | 0 | 3.817±0.581 | 2.290±1.405 | 9.160±1.355 | 9.160±1.317 |
| **19** | 0 | 0±0.581 | 0±1.405 | 0±1.355 | 0±1.317 |
| **20** | 0 | 3.704±0.581 | 16.296±1.405 | 16.296±1.355 | 16.270±1.317 |
| **21** | 0 | 3.145±0.581 | 16.981±1.405 | 18.868±1.355 | 18.868±1.317 |
| **22** | 0 | 8.8±0.581 | 14.4±1.405 | 15.2±1.355 | 16±1.317 |
| **23** | 0 | 0±0.581 | 0±1.405 | 0±1.355 | 0±1.317 |
| **24** | 0 | 1.987±0.581 | 24.503±1.405 | 21.854±1.355 | 13.907±1.317 |
| **25** | 0 | 7.826±0.581 | 27.826±1.405 | 26.523±1.355 | 26.522±1.317 |

# Appendix E – Individual Email Normalised KT Values for Test Sets

| Enron Formal Emails | Baseline 0 | Baseline 1 | Baseline 2 | Baseline 3 | Baseline 4 |
|---|---|---|---|---|---|
| **1** | 1 | 0.929±0.006 | 0.904±0.014 | 0.904±0.014 | 0.859±0.013 |
| **2** | 1 | 0.95±0.006 | 0.830±0.014 | 0.820±0.014 | 0.820±0.013 |
| **3** | 1 | 0.918±0.006 | 0.836±0.014 | 0.808±0.014 | 0.822±0.013 |
| **4** | 1 | 0.919±0.006 | 0.902±0.014 | 0.870±0.014 | 0.870±0.013 |
| **5** | 1 | 1±0.006 | 0.900±0.014 | 0.940±0.014 | 0.900±0.013 |
| **6** | 1 | 0.976±0.006 | 0.825±0.014 | 0.789±0.014 | 0.797±0.013 |
| **7** | 1 | 0.975±0.006 | 0.860±0.014 | 0.8599±0.014 | 0.860±0.013 |
| **8** | 1 | 0.912±0.006 | 0.824±0.014 | 0.824±0.014 | 0.824±0.013 |
| **9** | 1 | 0.962±0.006 | 0.798±0.014 | 0.827±0.014 | 0.846±0.013 |
| **10** | 1 | 0.940±0.006 | 0.900±0.014 | 0.884±0.014 | 0.900±0.013 |
| **11** | 1 | 0.948±0.006 | 0.835±0.014 | 0.814±0.014 | 0.804±0.013 |
| **12** | 1 | 0.939±0.006 | 0.849±0.014 | 0.849±0.014 | 0.818±0.013 |
| **13** | 1 | 0.944±0.006 | 0.889±0.014 | 0.889±0.014 | 0.889±0.013 |
| **14** | 1 | 0.973±0.006 | 0.920±0.014 | 0.926±0.014 | 0.941±0.013 |
| **15** | 1 | 1±0.006 | 0.962±0.014 | 0.962±0.014 | 0.962±0.013 |
| **16** | 1 | 0.991±0.006 | 0.947±0.014 | 0.939±0.014 | 0.965±0.013 |
| **17** | 1 | 0.956±0.006 | 0.884±0.014 | 0.844±0.014 | 0.856±0.013 |
| **18** | 1 | 0.962±0.006 | 0.977±0.014 | 0.908±0.014 | 0.908±0.013 |
| **19** | 1 | 1±0.006 | 1±0.014 | 1±0.014 | 1±0.013 |
| **20** | 1 | 0.963±0.006 | 0.837±0.014 | 0.837±0.014 | 0.837±0.013 |
| **21** | 1 | 0.969±0.006 | 0.830±0.014 | 0.811±0.014 | 0.811±0.013 |
| **22** | 1 | 0.912±0.006 | 0.856±0.014 | 0.848±0.014 | 0.840±0.013 |
| **23** | 1 | 1±0.006 | 1±0.014 | 1±0.014 | 1±0.013 |
| **24** | 1 | 0.980±0.006 | 0.755±0.014 | 0.781±0.014 | 0.861±0.013 |
| **25** | 1 | 0.922±0.006 | 0.722±0.014 | 0.735±0.014 | 0.735±0.013 |

| Personal Informal Emails | Baseline 0 | Baseline 1 | Baseline 2 | Baseline 3 | Baseline 4 |
|---|---|---|---|---|---|
| **1** | 1 | 0.978±0.006 | 0.822±0.014 | 0.844±0.017 | 0.844±0.017 |
| **2** | 1 | 0.971±0.006 | 0.892±0.014 | 0.912±0.017 | 0.912±0.017 |
| **3** | 1 | 0.955±0.006 | 0.773±0.014 | 0.636±0.017 | 0.636±0.017 |
| **4** | 1 | 0.983±0.006 | 0.803±0.014 | 0.777±0.017 | 0.768±0.017 |
| **5** | 1 | 0.981±0.006 | 0.870±0.014 | 0.759±0.017 | 0.759±0.017 |
| **6** | 1 | 0.873±0.006 | 0.825±0.014 | 0.810±0.017 | 0.810±0.017 |
| **7** | 1 | 0.899±0.006 | 0.889±0.014 | 0.909±0.017 | 0.909±0.017 |
| **8** | 1 | 0.976±0.006 | 0.827±0.014 | 0.827±0.017 | 0.819±0.017 |
| **9** | 1 | 0.975±0.006 | 0.95±0.014 | 0.938±0.017 | 0.9375±0.017 |
| **10** | 1 | 0.915±0.006 | 0.854±0.014 | 0.805±0.017 | 0.805±0.017 |
| **11** | 1 | 0.969±0.006 | 0.938±0.014 | 0.938±0.017 | 0.938±0.017 |
| **12** | 1 | 0.958±0.006 | 1±0.014 | 1±0.017 | 1±0.017 |
| **13** | 1 | 0.899±0.006 | 0.870±0.014 | 0.870±0.017 | 0.870±0.017 |
| **14** | 1 | 0.924±0.006 | 0.857±0.014 | 0.882±0.017 | 0.882±0.017 |
| **15** | 1 | 0.939±0.006 | 0.960±0.014 | 0.973±0.017 | 0.973±0.017 |
| **16** | 1 | 1±0.006 | 0.940±0.014 | 0.833±0.017 | 0.833±0.017 |
| **17** | 1 | 0.945±0.006 | 0.881±0.014 | 0.706±0.017 | 0.706±0.017 |
| **18** | 1 | 0.915±0.006 | 0.930±0.014 | 0.803±0.017 | 0.803±0.017 |
| **19** | 1 | 1±0.006 | 0.921±0.014 | 0.921±0.017 | 0.921±0.017 |
| **20** | 1 | 0.964±0.006 | 0.838±0.014 | 0.874±0.017 | 0.874±0.017 |
| **21** | 1 | 0.943±0.006 | 0.836±0.014 | 0.898±0.017 | 0.885±0.017 |
| **22** | 1 | 0.965±0.006 | 0.883±0.014 | 0.887±0.017 | 0.887±0.017 |
| **23** | 1 | 0.938±0.006 | 0.647±0.014 | 0.905±0.017 | 0.917±0.017 |
| **24** | 1 | 0.947±0.006 | 0.824±0.014 | 0.807±0.017 | 0.807±0.017 |
| **25** | 1 | 0.915±0.006 | 0.830±0.014 | 0.756±0.017 | 0.756±0.017 |

| Personal Formal | Baseline 0 | Baseline 1 | Baseline 2 | Baseline 3 | Baseline 4 |
|---|---|---|---|---|---|

| **Emails** | | | | | |
|---|---|---|---|---|---|
| **1** | 1 | 0.955±0.006 | 0.8310.830±0.019 | 0.864±0.021 | 0.867±0.021 |
| **2** | 1 | 1±0.006 | 1±0.019 | 1±0.021 | 1±0.021 |
| **3** | 1 | 0.967±0.006 | 0.967±0.019 | 0.956±0.021 | 0.901±0.021 |
| **4** | 1 | 0.934±0.006 | 0.893±0.019 | 0.889±0.021 | 0.885±0.021 |
| **5** | 1 | 0.930±0.006 | 0.940±0.019 | 0.909±0.021 | 0.909±0.021 |
| **6** | 1 | 1±0.006 | 1±0.019 | 1±0.021 | 1±0.021 |
| **7** | 1 | 1±0.006 | 0.723±0.019 | 0.766±0.021 | 0.787±0.021 |
| **8** | 1 | 0.937±0.006 | 0.779±0.019 | 0.747±0.021 | 0.747±0.021 |
| **9** | 1 | 0.906±0.006 | 0.943±0.019 | 0.943±0.021 | 0.943±0.021 |
| **10** | 1 | 0.958±0.006 | 0.831±0.019 | 0.761±0.021 | 0.761±0.021 |
| **11** | 1 | 0.972±0.006 | 0.778±0.019 | 0.764±0.021 | 0.729±0.021 |
| **12** | 1 | 0.973±0.006 | 0.946±0.019 | 0.919±0.021 | 0.919±0.021 |
| **13** | 1 | 0.916±0.006 | 0.863±0.019 | 0.832±0.021 | 0.855±0.021 |
| **14** | 1 | 0.938±0.006 | 0.870±0.019 | 0.880±0.021 | 0.880±0.021 |
| **15** | 1 | 0.934±0.006 | 0.851±0.019 | 0.818±0.021 | 0.818±0.021 |
| **16** | 1 | 0.927±0.006 | 0.824±0.019 | 0.832±0.021 | 0.815±0.021 |
| **17** | 1 | 0.948±0.006 | 0.814±0.019 | 0.835±0.021 | 0.832±0.021 |
| **18** | 1 | 0.954±0.006 | 0.806±0.019 | 0.749±0.021 | 0.749±0.021 |
| **19** | 1 | 0.898±0.006 | 0.878±0.019 | 0.857±0.021 | 0.755±0.021 |
| **20** | 1 | 0.928±0.006 | 0.717±0.019 | 0.651±0.021 | 0.651±0.021 |
| **21** | 1 | 1±0.006 | 1±0.019 | 1±0.021 | 1±0.021 |
| **22** | 1 | 0.969±0.006 | 0.781±0.019 | 0.766±0.021 | 0.766±0.021 |
| **23** | 1 | 1±0.006 | 1±0.019 | 1±0.021 | 1±0.021 |
| **24** | 1 | 0.971±0.006 | 0.647±0.019 | 0.647±0.021 | 0.647±0.021 |
| **25** | 1 | 0.953±0.006 | 0.901±0.019 | 0.896±0.021 | 0.896±0.021 |

| Enron Informal Emails | Baseline 0 | Baseline 1 | Baseline 2 | Baseline 3 | Baseline 4 |
|---|---|---|---|---|---|
| **1** | 1 | 0.921±0.005 | 0.869±0.013 | 0.867±0.013 | 0.851±0.013 |
| **2** | 1 | 0.935±0.005 | 0.855±0.013 | 0.889±0.013 | 0.880±0.013 |
| **3** | 1 | 1±0.005 | 0.688±0.013 | 0.688±0.013 | 0.688±0.013 |
| **4** | 1 | 0.973±0.005 | 0.870 ±0.013 | 0.860±0.013 | 0.860±0.013 |
| **5** | 1 | 0.949±0.005 | 0.836±0.013 | 0.836±0.013 | 0.836±0.013 |
| **6** | 1 | 0.954±0.005 | 0.840±0.013 | 0.809±0.013 | 0.779±0.013 |
| **7** | 1 | 0.930±0.005 | 0.842±0.013 | 0.930±0.013 | 0.930±0.013 |
| **8** | 1 | 0.949±0.005 | 0.863±0.013 | 0.846±0.013 | 0.846±0.013 |
| **9** | 1 | 0.976±0.005 | 0.812±0.013 | 0.773±0.013 | 0.773±0.013 |
| **10** | 1 | 0.946±0.005 | 0.919±0.013 | 0.892±0.013 | 0.892±0.013 |
| **11** | 1 | 0.909±0.005 | 0.818±0.013 | 0.805±0.013 | 0.805±0.013 |
| **12** | 1 | 0.984±0.005 | 0.929±0.013 | 0.738±0.013 | 0.738±0.013 |
| **13** | 1 | 0.990±0.005 | 0.790±0.013 | 0.840±0.013 | 0.840±0.013 |
| **14** | 1 | 0.975±0.005 | 0.951±0.013 | 0.877±0.013 | 0.877±0.013 |
| **15** | 1 | 0.920±0.005 | 0.898±0.013 | 0.920±0.013 | 0.920±0.013 |
| **16** | 1 | 0.985±0.005 | 0.938±0.013 | 0.800±0.013 | 0.800±0.013 |
| **17** | 1 | 0.948±0.005 | 0.935±0.013 | 0.922±0.013 | 0.922±0.013 |
| **18** | 1 | 0.937±0.005 | 0.857±0.013 | 0.837±0.013 | 0.841±0.013 |
| **19** | 1 | 0.986±0.005 | 0.859±0.013 | 0.803±0.013 | 0.803±0.013 |
| **20** | 1 | 0.945±0.005 | 0.868±0.013 | 0.845±0.013 | 0.845±0.013 |
| **21** | 1 | 0.929±0.005 | 0.865±0.013 | 0.810±0.013 | 0.817±0.013 |
| **22** | 1 | 0.958±0.005 | 0.806±0.013 | 0.819±0.013 | 0.819±0.013 |
| **23** | 1 | 0.950±0.005 | 1±0.013 | 1±0.013 | 1±0.013 |
| **24** | 1 | 0.946±0.005 | 0.806±0.013 | 0.793±0.013 | 0.793±0.013 |
| **25** | 1 | 0.958±0.005 | 0.875±0.013 | 0.875±0.013 | 0.813±0.013 |

# Appendix F – Attachments

The attached USB contains:

- Source code of software
    - Baseline3.py
    - Clean_bnc.py
- Sample corpora text files for training model
    - Formal_Emails_Filtered_All.txt
    - Informal_Emails_Filtered_All.txt
    - Past_PDFs.txt
    - User_based_corpus.txt
- Note: Due to distribution policies, the BNC needs to be accessed externally