Saarland University

Department of Computational Linguistics and Phonetics

Master's Thesis

Ranking Sentences by Complexity

Submitted in partial fulfillment of the requirements for the degree
Master of Science in Language Science and Technology.

Author:
David M. Howcroft
howcroft@coli.uni-saarland.de

Supervisors:
Dr. Vera Demberg
Dr. Matthew Crocker

December 2015

To my parents, Kevin and Mary-Louise Howcroft

Declaration

I hereby confirm that the thesis presented here is my own work, with all assistance acknowledged.

_____ , _____ _____

Place Date Signature

Abstract

Work in readability has typically focused on document-level measures of text difficulty, but many natural language generation applications would benefit from improved models of text difficulty for individual sentences. Fortunately, research in psycholinguistics on the processing of sentences by human readers has yielded promising features, such as surprisal (Hale, 2001), integration cost (Gibson, 2000), embedding depth (van Schijndel and Schuler, 2012; van Schijndel et al., 2013), and idea density (Kintsch, 1972; Kintsch and Keenan, 1973).

In this thesis, I evaluate the effectiveness of these features for ranking sentences by their complexity. Using sentence-aligned corpora drawn from Wikipedia (Hwang et al., 2015, ESEW) and edited news texts (Vajjala, 2015, OSE) as a source of labeled sentence pairs (ESEW) and triples (OSE), I train a series of linear models based on these psycholinguistic features.

My results show that psycholinguistic features improve model accuracy significantly over a simple baseline using word length and sentence length features, with gains in the 0.5 to 3 percentage point range. However, overall model accuracy remains in the 70-80% range, suggesting that these features may not be worth the extra time required to parse sentences using a cognitively-plausible parser.

Acknowledgements

I would like to thank my supervisors, Vera Demberg and Matthew Crocker, who have offered good advice along the way as well as guidelines for completing this work in a timely fashion. A thesis is an uncanny beast to tame, and it requires some reassurance that you are heading down the right path. Vera and Matt have both provided this in some measure and, in so doing, made this thesis possible.

My early thoughts on this topic received a lot of feedback from Michael White and Eric Fosler-Lussier at Ohio State, and I am indebted to audiences at the Clippers discussion group and the Department of Linguistics there more generally for their helpful feedback. I had many good conversations with Marten van Schijndel and William Schuler on the topic of sentence complexity and also found their help invaluable when it came time to use the `ModelBlocks` parser.

More recently, Detmar Meurers and Sowmya Vajjala were kind enough to invite me to Tübingen to speak at Detmar's Oberseminar on 'Linguistic Modelling and its Interfaces'. His group had many good comments and questions and Sowmya and I were able to make plans for future collaboration in this area. Sowmya was also kind enough to provide the most recent version of her One Stop English corpus for my use, in addition to our many fruitful conversations.

Closer to home, Lisa Teunissen and Marc Schulder read sections of this thesis in a frightful stage of development with good humor and my writing is stronger for their suggestions. Naturally, I would also like to thank my officemates and other colleagues not mentioned by name for their input and support (and patience) throughout these past several months.

And I would not be here if not for Kevin and Mary-Louise Howcroft (my parents) and Alyssa Price, my partner and my rock.

Funding

Initial work in this direction was supported in part by a Targeted Investment in Excellence grant to the Ohio State University Department of Linguistics During the thesis itself I was supported by the Deutsche Forschungsgemeinschaft (DFG) through Sonderforschungsbereiche (SFB) 1102: Information Density and Linguistic Encoding.

Contents

List of Figures

List of Tables

1 Introduction

Reading is an acquired skill, with all the variation that entails. Some texts are more difficult than others. Readers also differ in terms of their ability to read. Second language learners, adult literacy students, and individuals coping with specific medial problems, such as aphasia, all stand to benefit from advances in automatic text simplification.

The last twenty years have seen a variety of work in this direction, ranging from rule-based systems targeting specific populations (Chandrasekar et al., 1996; Siddharthan, 2004) to the general-purpose simplification-as-monolingual-statistical-machine-translation approaches of recent years (Zhu et al., 2010, inter alia). Despite their varied scope, these projects have generally avoided the problem of modeling text difficulty explicitly. An explicit model of sentence difficulty, however, would serve both to identify sentences requiring simplification and to confirm that a given simplification system achieves its intended goal, producing a simpler sentence.

During the same time period there has been a great deal of work on understanding *human* sentence processing (i.e. reading). This has fostered the development of theories of sentence processing rooted in both linguistic theory and psychological experimentation. For example, eye-tracking studies of reading times have found that measures such as *integration cost* and *surprisal* provide partial explanations for subjects' reading behavior (Demberg and Keller, 2008).

This thesis leverages the features suggested by these psycholinguistic theories of sentence processing in order to improve our ability to score individual sentences by their difficulty. Such a scoring mechanism will serve the needs of the text simplification community, helping to identify sentences that should be simplified and, indeed, verifying that such systems manage to 'simplify' their input sentences. This second ability is also more generally useful in the field of natural language generation (NLG) in any application where a text is targeted to a particular difficulty level.

With these goals in mind, we aim to test one main hypothesis: that incorporating psycholinguistically-motivated features relating to sentence processing difficulty into a sentence ranking task will improve the performance of a baseline model of sentence difficulty. In the course of testing this hypothesis, we will also examine the contributions of the individual features to the model and test the utility of these features in examining fine-grained (as opposed to coarse-grained) distinctions in sentence difficulty.

1.1 Structure of this thesis

After initial background on readability and existing measures of text difficulty (Chapter 2), we review the psycholinguistic theories under consideration (Chapter 3). We describe two related machine learning approaches for ranking (Chapter 4) and explain the methodology of our current experiments (Chapter 5). We present our analysis in Chapter 7, before discussing future work in our concluding thoughts (Chapter 8). Several appendices contain further material pertinent to our methodology and analyses.

2 Readability

Readability is often mentioned without a clear definition and can be reasonably defined in a variety of ways. Is the readability of a text connected to the legibility of its presentation? the reader's success at extracting information from the text? the ease with which the text is read? For the purposes of this study, however, we limit the scope of readability to that of reading ease, focusing on linguistic constraints rather than constraints of medium (relating to e.g. legibility) or reader interest. Our assumption is that making material easier to read will also make it easier to comprehend.

In this chapter we review the origins of modern work on readability before discussing several more recent studies on the topic. The contribution of this thesis lies in our choice of features for modeling readability (i.e. text complexity) and our focus on the readability of individual sentences.

2.1 Early readability research

The most comprehensive review of readability research in the first half of the twentieth century is that of Chall (1958), who is known for her part in developing the Dale-Chall readability formula. She divides the early work in readability into two main categories: "survey and experimental studies" and "quantitative associational studies". Studies of the former category took place during the 1930s and 1940s and included surveys of expert and reader opinion as well as experimental studies which manipulated texts according to one variable at a time in order to determine the effects of those variables on readers. The results of these studies suggest that, once you have managed to control for reader interest in the content of a text, the most important factor with respect to its readability is its 'style', e.g. its "scope

of vocabulary and...kinds of sentences" (Gray and Leary, 1935, as quoted in (Chall, 1958)).

The second class of study is more akin to our own, correlating features of a text with its ordering relative to some other texts. The earliest of these studies according to Chall is from Lively & Pressey (1923), which primarily focused on vocabulary as a bottleneck in science education. Not noted by Chall, however, was the work from thirty years earlier by L. A. Sherman, who proposed a quantitative analysis of text difficulty for pedagogical purposes (Sherman, 1893). While Sherman approached the subject from an empirical perspective, lauding science as "in its method of substituting experiments and experiences for second-hand knowledge...[finding] a means of bridging the chasm between exceptionally endowed and mediocre minds", his studies were neither comprehensive nor systematically tested and reviewed. On the basis of 500-word excerpts of texts from a number of authors, Sherman proposed that the difficulty students would have with a text could be predicted by its average sentence length in words and the number of clauses in each sentence, making him the first to propose a quantitative measure of text difficulty.

These varied studies on quantifying readability culminated in the mid-twentieth century with a number of readability formulae[1], including the familiar Flesch-Kincaid Grade-Level score (Kincaid et al., 1975, FKGL), shown in 2.1.

$$0.39\frac{\#\dot{w}ords}{\#sentences} + 11.8 \cdot \frac{\#syllables}{\#words} - 15.5 \tag{2.1}$$

These formulae typically use a linear combination of average word length and average sentence length. Initially it was not uncommon to measure word-length in syllables, as in the Flesch Reading Ease formula (Flesch, 1948), when human editors were manually estimating readability to calculate reading difficulty. Later measures, like the Automated Readability Index (Smith and Senter, 1967, ARI) sometimes measured word length in characters, enabling automatic counting by by electronic typewriters. Sentence length is typically measured in white-space delimited words, not counting any punctuation that might be isolated by tokenization.

[1]For a comprehensive review of the literature up to 1958, we recommend (Chall, 1958). For a more recent review of the literature, we recommend Chapter 2 of (Vajjala, 2015). For an introduction to some of the major studies from the 20th century, we recommend the self-published (Dubay, 2007).

Some of these formulae incorporate a vocabulary-diversity term. For example, the Dale–Chall readability formula incorporates a measure of the proportion of words not appearing on a list of frequent words (Dale and Chall, 1948). Measures of this kind are based on the observation that familiar words should be easier to process while rarer words and jargon will generally increase the difficulty of a text. Indeed, a text too dense with jargon may become entirely impenetrable.

While work in the second half of the 20th century began to focus on automating the process of estimating readability (e.g. the aforementioned ARI score), the early formulae were limited to these simple linear models by practical necessity, as feature extraction had to be performed manually. More recent work has sought to apply modern tools for natural language processing and machine learning to the problem, although the focus has continued to be on evaluating whole documents rather than their sub-parts.

2.2 Recent Work in Readability

Petersen (2007) took up the problem of text leveling for second-language learners using techniques common in NLP. In particular, she used a number of parse-based features which captured, for example, the average number of noun and verb phrases per sentence and the height of the parse tree. Petersen trained SVM classifiers to classify texts as belonging to one of four primary school grade levels based on the Weekly Reader educational newspaper[2]. These document-level models achieved F-scores in the range of 0.5 to 0.7, compared to the F-scores between 0.25 and 0.45 achieved by the Flesch-Kincaid Reading Ease score for the same texts.

Petersen was also interested in automatic text simplification, but focused primarily on when a sentence should be split into multiple sentences or dropped completely during simplification. Therefore she trained a decision tree model to predict when a sentence should be split or deleted, but did not extend her readability models to individual sentences.

In addition to these lexical and syntactic features, recent work has looked at cognitively-motivated measures related to discourse. Feng et al. (2009) worked on a model of readability for adults with intellectual disabilities. Their target pop-

[2] `http://www.weeklyreader.com`

ulation led them to consider factors which might be influenced by working memory deficits—specifically, the number of entities mentioned in a document. They found that their cognitively-motivated features resulted in a better correlation (Pearson's $R = -0.352$) compared to both Flesch-Kincaid score ($R = -0.270$) and a number of 'basic' linguistic features based on those used by Petersen & Ostendorf (2009) ($R = -0.283$).[3]

Other discourse-level text features have also been incorporated into readability models in recent years. Coh-Metrix (Graesser et al., 2004) includes a number of measures related to discourse coherence, for example. Such features are not suited to the problem of determining the difficulty of sentences in isolation, but they have also been shown to better predict readability for second-language learners compared to 'traditional' readability measures like those described in §2.1 (Crossley et al., 2011).

This sampling of studies illustrates the breadth of linguistic features incorporated into recent models of readability, including features at the lexical, syntactic, semantic, and discourse levels. Another recurring theme is that each model is usually tailored to a particular audience. While traditional measures were often intended for general use by adult native readers of English, they were sometimes refined for specific target populations, as when ARI, Fog Count, and the Flesch Reading Ease formulae were refined for use by the United States Navy in (Kincaid et al., 1975). Recent work has also emphasized models of readability for second-language learners and readers with disabilities.

2.3 Sentence-level Readability

Few studies to date have addressed sentence-level readability for English. Napoles & Dredze (2010) used documents from English and Simple English Wikipedia to train both document- and sentence-level classifiers. Using unigram bag-of-words features, unigram and bigram part-of-speech features, type-token ratio, the proportion of words appearing on a list of easier words, and parse features similar to those of Petersen (see §2.2), their SVM classifier achieved an accuracy of about 77% on this

[3] Correlations here are negative because Feng et al. were correlating their predicted reading levels with the performance of adults with intellectual disabilities on comprehension tests based on a set of leveled texts. The adults with disabilities are expected to perform worse on the comprehension test as the grade level of the text increases.

task. This accuracy seems quite reasonable considering that sentences in English Wikipedia can also be simple, so some amount of classification error is inevitable.

In the medical domain, Kauchak et al. (2014) also looked at sentence-level classification, identifying sentences as being either simple or difficult. Their features included word length, sentences length, part-of-speech counts, average unigram frequencies and standard deviation, and the proportion of words not on a list of the five thousand most frequent words as well as three domain-specific features based on an ontology of medical terminology.

The only line of work *ranking* instead of *classifying* sentences is that of Vajjala & Meurers (Vajjala and Meurers, 2014; Vajjala, 2015). Vajjala & Meurers initially approached the problem by training a readability model on levelled documents. These documents fell into five different categories corresponding to the levels of the WeeBit corpus (Vajjala and Meurers, 2012). This model correlated quite well (Spearman's $\rho = 0.69$) with a second corpus consisting of levelled texts from the Common Core Standards (CCSSO, 2010), fairly typical for readability formulae.

Moving on to sentence classification, however, they found that applying the same model to individual sentences from English and Simple English Wikipedia resulted in an accuracy of only 66%, considerably lower than the 77% of Napoles & Dredze (2010). This makes sense when considering their dataset, however. While we might expect a model to learn which of a pair of sentences is more difficult, there is actually a good deal of overlap between articles from English and Simple English Wikipedia: in machine learning terms, these datasets do not form separable classes. While not every article in Simple Wikipedia is a "translation" of an article from English Wikipedia, those that are are *relatively* simplified—i.e. they are simplified with respect to some reference sentence. The difficulty of the reference sentence from English Wikipedia, then, partially determines the difficulty of the Simple English sentence, such that a sentence from Simple English Wikipedia, while simpler than its source sentence, may in fact be harder than another sentence chosen from English Wikipedia.

Indeed, Vajjala & Meurers (2014) report such an overlap in their Figure 2, reproduced here as Figure 2.1. This graph shows the distribution across grade levels for each portion of the Parallel Wikipedia corpus (Zhu et al., 2010, PWKP), as predicted by their grade-level (multi-class) classification model trained on the Wee Bit corpus. While the Simple English portion of the corpus is clearly skewed toward the lower grade levels, it appears that the English portion of the corpus is fairly evenly

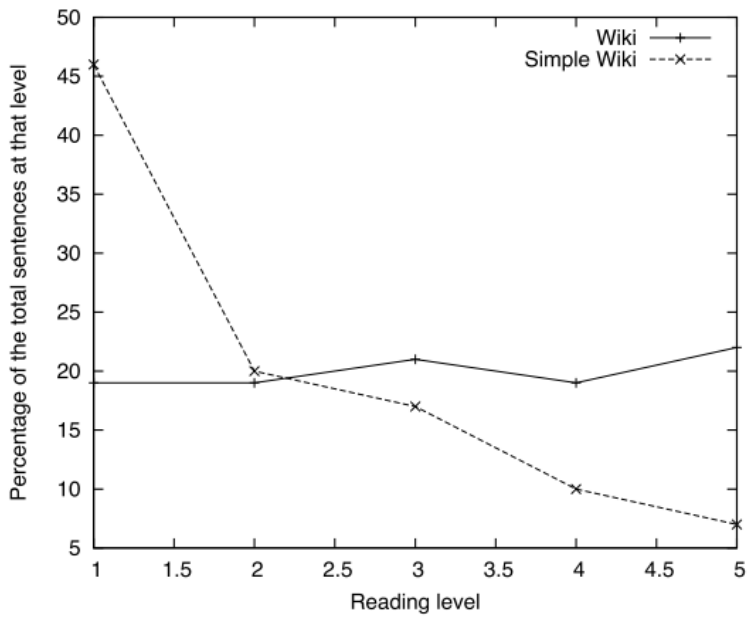

Figure 2.1: Reading levels in English and Simple English Wikipedia, as estimated by Vajjala & Meurers (2014) and shown in that paper.

distributed across all grade levels, making any binary-classification-based approach more difficult.

In response to this problem, Vajjala & Meurers used these predicted reading levels from their multiclass classifier to rank pairs of aligned sentences relative to each other. To evaluate this ranking, each pair of aligned sentences was evaluated based on the accuracy of their predicted labels with respect to the ranking implied by the assumption that, given a pair of aligned sentences from English and Simple English Wikipedia, the sentence from Simple English Wikipedia should be less difficult. Using a hard cut-off (i.e. $rank(EW) > rank(SEW)$), their model achieved about 59% accuracy.[4] Counting sentence pairs predicted to belong to the same reading level as correctly ranked, accuracy rises to 70%. Allowing this equality test to include sentences which differ by up to one level (on the five level scale) as being correctly ranked leads to a 90% accuracy level.

Vajjala's (2015) dissertation follows up on this work, again examining the PWKP corpus but from an explicitly ranking-oriented approach. After testing a few machine

[4]Note, however, that this is not strictly comparable to the 66% accuracy on the binary classification task reported above. Indeed, the probability of any pair of sentences being accurately classified in such a model (assuming independence) would be about 44%, lower even than chance.

learning techniques for ranking with the same set of features, she found that SVM^{rank} performed the best. She also created the OSE corpus, which we describe in §5.1 and use for our own study. Explicitly addressing the ranking task, Vajjala (2015) achieved an 82.7% accuracy (std. dev. 8.4 percentage points) on the PWKP corpus using 10-fold cross validation and 79.7% on held-out test data.

2.3.1 Features used by Vajjala & Meurers

It is worth describing the features of Vajjala & Meurers (2012; 2014) in more depth in order to situate this thesis properly. Vajjala & Meurers used a variety of linguistic and psycholinguistic features in their readability models, dividing them roughly into lexical, syntactic, traditional, and psycholinguistic features.

Lexical features included n-grams, part-of-speech tags and POS-tag ratios, various forms of Type-Token Ratio, measures of lexical variation and lexical density, word length in syllables and number of characters, and the proportion of words appearing on a list of academic words. These are supplemented with word frequency and morphological information from the CELEX database (Baayen et al., 1995).

Syntactic features included features based on parse trees such as the average parse tree height and the average lengths of different XPs. They also incorporated measures based on the notion of the *T-unit* from the second language acquisition literature.

While sentence length and word length features were incorporated into the syntactic and lexical feature categories, respectively, the **Traditional** features category further included two measures from the document-level readability literature: Flesch-Kincaid score (Kincaid et al., 1975) and Coleman-Liau readability score (Coleman and Liau, 1975).

Psycholinguistic features come from the MRC psycholinguistic database (Wilson, 1988) and include word familiarity, concreteness, imageability, meaningfulness, and age of acquisition. These features were coupled with a second age of acquisition database and values related to the average number of senses per word.

Of these features, ours are most similar to those in the syntactic category. It remains to be seen, however, to what extent our features might subsume theirs, or vice versa.

The relative complementarity of Vajjala & Meurer's psycholinguistic features with our own suggests that a model combining these sets of features may see further improvement.

2.3.2 Relevance to this thesis

Based on the finding that 30% of sentence pairs from the PWKP corpus are incorrectly ranked despite lying within one reading level of each other, we hypothesize that finer-grained distinctions may be necessary to tease apart the differences in related pairs of sentences. To this end, this thesis leverages psycholinguistic measures that have been found to correlate with reading times in eye-tracking and self-paced reading studies. The particular features we examine are detailed now in §3.

3 Psycholinguistic Theories of Sentence Processing

Psycholinguistic research aims to understand how humans comprehend and produce natural language utterances and texts. A subset of this work, then, focuses on the processing of individual sentences during reading. In this chapter we discuss four theoretical measures proposed to account for sentence processing difficulty and the experimental evidence that these measures succeed in their goal. In particular, we survey surprisal, embedding depth, integration cost, and idea density.

3.1 Surprisal

Surprisal is a measure of how surprising or informative a word is given its context (Shannon, 1948). The context for a word is usually operationalized as the probability of the preceding sequence of words, as in:

$$surprisal(w_n) = -log(P(w_n|\textsc{context})) \tag{3.1}$$

$$= -log(P(w_n|w_1 \dots w_{n-1})) \tag{3.2}$$

where w_i is the ith word in the sentence and $P(w_1 \dots w_i)$ denotes the probability of the sequence of i words $w_1 \dots w_i$.

The simplest way of approximating the probability of the preceding word sequence is using n-grams. In this approach we limit the preceeding context to one or two words (resulting in *bigram* and *trigram* probabilities, respectively). This results in the loss of some information as, e.g., we approximate $P(w_4|w_1w_2w_3)$ as $P(w_4|w_2w_3)$,

but data sparsity (i.e. the low counts of n-grams for larger values of n) requires such approximation.

These probabilities can be more comprehensively calculated, however, using a probabilistic parser (Hale, 2001) which explicitly takes into account all possible syntactic structures compatible with the observed word sequence. In particular, the probability of a sequence $w_1 \dots w_n$ is the sum of the probabilities of all possible probabilistic context free grammar (PCFG) trees having the sequence $w_1 \dots w_n$ at their leaves. Under this view, (total) surprisal is usually taken as the sum of two components:

- *syntactic* surprisal, or the surprisal score based on the parse up to and including the pre-terminal node (i.e. the syntactic category for the current word); and
- *lexical* surprisal, or the surprisal score based on the probability of the current word given this syntactic structure.

That is, if we have a set of derivations $D(w_1 \dots w_n)$ for a sequence of words $w_1 \dots w_n$ where the last step in any derivation $D[|D|-1, |D|]$ is to generate word w_n, we have the following (Roark et al., 2009):

$$
\begin{aligned}
surprisal(w_n) &= -log\frac{\Sigma_{D(w_1 \dots w_n)} p(D)}{\Sigma_{D(w_1, wn-1)} p(D)} && (3.3)\\
&= -log\frac{\Sigma_{D(w_1 \dots w_n)} p(D[1, |D|-1])}{\Sigma_{D(w_1, wn-1)} p(D)} && (3.4)\\
& \quad -log\frac{\Sigma_{D(w_1 \dots w_n)} p(D)}{\Sigma_{D(w_1 \dots w_n)} p(D[1, |D|-1])} && (3.5)\\
&= \textsc{SyntacticSurprisal} + \textsc{LexicalSurprisal} && (3.6)
\end{aligned}
$$

Levy (2008) argues that surprisal models language users as rational learners. The rational reader's attention or cognitive resources must be spread across all possible analyses for the sentence being observed. Based on previous experience the reader expects some analyses to be more probable than others and therefore allocates more resources to those analyses. Thus surprisal is directly derivable as a measure of the cost paid when the reader misallocates resources: when a new word invalidates one or more highly probable analyses, the reader has effectively 'wasted' whatever resources were allocated to those analyses.

The *Uniform Information Density* hypothesis (Jaeger, 2006; Levy and Jaeger, 2007; Jaeger, 2010, UID) examines these costs in the context of Shannon's (1948) *noisy channel coding theorem.*[1] These costs are especially dire when they exceed a listener's current cognitive capacity, modelled as the *channel capacity.* Furthermore, under-exploitation of the channel between the speaker and the listener results in inefficiencies, whereby the speaker is exerting more effort than necessary to communicate their point.

In terms of reading sentences, this means that a sentence will be read efficiently when the surprisal at each word in the sentence is near the channel capacity without exceeding it. Peaks in surprisal above the capacity of the channel are proposed to result in extreme slowdowns as the reader must re-evaluate their current analysis of the sentence. This provides further theoretical motivation for the importance of surprisal as a measure of sentence processing difficulty.

These theoretical claims are borne out by Demberg & Keller's (2008) study of the relationship between surprisal and eye-tracking times in the Dundee corpus (Kennedy and Pynte, 2005) Demberg & Keller found that increased surprisal significantly correlated with reading times. These results are corroborated by the self-paced reading study of Wu et al. (2010).

Another eye-tracking study divided surprisal into lexical and synactic components as a refinement of these approaches (Roark et al., 2009). Roark et al. found a significant effect for lexical surprisal, but not for syntactic surprisal, thus justifying the importance of distinguishing these measures in our study of sentence difficulty.

3.2 Embedding Depth

Yngve (1960) proposed a mechanistic incremental model of language production and examined how phrase structure rule (PSR) grammars could be evaluated with respect to their expected processing complexity for a computing system with finite memory. Based on Miller's (1956a; 1956b) notion that human working memory is limited to 7 ± 2 items, Yngve explores what structure speakers must keep track

[1] Note as well that (Shannon, 1948, p. 5, para. 1) is the original source of information theory in general and surprisal measures in particular.

of during production and how grammars might be structured to avoid overtaxing working memory.

The mechanism Yngve describes uses a finite PSR grammar, stored unordered in long-term memory. This grammar is then used for incremental, word-by-word production, proceeding by left-to-right expansion of rules drawn from the PSR. Constituents awaiting expansion are stored in temporary memory, which is taken to be analogous to human working memory and is therefore restricted to be finite.

On the basis of this framework, he concludes that well-behaved PSR grammars should favor binary, right-branching structures and avoid what he calls 'ungrammatical first steps', or structures where a rule can recursively apply in a left-branching fashion. In order to avoid high memory load, a grammar might include restricted relabelling, wherein applications of a rule expansion are labeled with their nestedness and it is ungrammatical to apply a rule with too high a nestedness value. Possibly alternatively, a grammar would prefer structural reversal to shift left-branching structures to the right where possible and prefer to allow for 'discontinuous constituents' to capture phenomena like heavy noun phrase (NP) shift and other forms of structural overlap.

These ideas are taken up in van Schijndel et al. (2012; 2013), who model the minimal number of memory states required to parse a sentence using a hierarchical sequence model of parsing. Consider an incremental model of sentence processing where, at each stage, the reader has an *active* state (e.g. S for sentence) and an *awaited* state (e.g. VP for verb phrase). For example, the state described above (i.e. that of being in the active state S and awaiting state VP) might be reached after a reader has observed a noun phrase, resulting in the state S/VP. This means that the word sequence observed so far will be consistent with a sentence if we now observe a verb phrase. If, however, the next word in the input is inconsistent with the start of a verb phrase (e.g. the relative clause marker *that*), then this parse will be ruled out and another must be considered.

Upon encountering *that*, the parser can hypothesize the start of a new *connected component*, i.e. a new syntactic substructure that must be completed before 'returning' to the parsing of the top-level of the sentence. That is, the parser must now keep track of two states: (1) the fact that we are still looking for a VP to complete the overall sentence and (2) the fact that we now have a relative clause to parse before we can complete the current NP. These connected components are the pars-

ing analogues to the constituents awaiting expansion stored in Yngve's temporary memory.

These measures provide an idealized representation of the number of different states a human parser must keep track of at any point in time. We refer to this number of states as the *embedding depth* of a sentence at a particular word, and the `ModelBlocks` parser of van Schijndel et al. (2012) calculates this number of states averaged over the beam of currently plausible parses. Also of interest is the *embedding difference*, which is the embedding depth at the present word relative to the previous word.[2]

Wu et al's (2010) study of psycholinguistic complexity metrics also included embedding depth as a feature. Their study of the reading times of 23 native English speakers reading four narratives indicated that embedding difference was a significant predictor of reading times for closed class words. Moreover, this contribution was independent of the contribution of surprisal, indicating that the two measures are capturing different components of the variance in reading times.

3.3 Integration Cost

Syntactic dependencies are binary relations which hold between the words of a sentence. Each relation has exactly one head and one dependent, and every word is the dependent of exactly one word and may be the head for one or more other words, except for the word which is the root of the dependency tree and therefore is only a head.

The length of a dependency can be measured in a number of different ways. Gibson (2000), in his Dependency Locality Theory (DLT), proposes that the relevant unit of distance is the number of intervening discourse referents. This model maintains that the act of creating a new discourse referent and holding it in memory makes it more difficult to recall a previous discourse referent and connect that discourse referent to the current one.

With its focus on discourse referents, Gibson's DLT defines a measure of *integra-*

[2] In our example above, we are at embedding depth 1 or 0 up until we encounter the word *that*, which increases the embedding depth by 1, resulting in a nonzero embedding difference score.

tion cost primarily for nouns and verbs. In contrast, Gildea & Temperley (2010) measure dependencies in terms of word span, such that adjacent words have a dependency length of one. This approach behaves similarly to integration cost for nouns and verbs, with the caveat that distances are systematically increased, and has the advantage of being defined for all words in a sentence.

Demberg & Keller (2008) found that DLT's integration cost did not contribute significantly to predicting eye-tracking reading times in general, although its contribution is significant when restricted to nouns and verbs. They also found that surprisal and integration cost were uncorrelated, suggesting that they should be considered complementary factors in a model of reading times.

As embedding difference has been demonstrated to be a significant predictor of reading times for closed-class words (i.e. not nouns and verbs) (Wu et al., 2010), integration cost and embedding depth should also be complementary.

3.4 Idea Density

Keenan & Kintsch (1973) looked at the influence of *propositional idea density* on reading times and recall. Using the propositional base (Kintsch, 1972) for a series of texts, they calculated the ratio of propositions or ideas to words in the sentences presented to subjects. This notion of idea density is closely related to Perfetti's (1969) notion of *lexical density* insofar as both are related to the number of so-called *content words* in the text.

Keenan & Kintsch conducted two different experiments in order to examine free reading behavior as well as subjects' performance in speeded reading conditions. They found that "the number of propositions [in a text] had a large effect upon reading times, [but] it could only account for 21% of their variance" when subjects were allowed to read freely. Subjects' overall recall was worse for more dense texts in the speeded reading condition. In addition to effects of idea density, they found that propositions which were presented as surface-form modifiers (as opposed to, e.g., main verbs) were "very poorly recalled" and that propositions playing a subordinate role relative to another proposition were also less-well recalled. Finally, propositions involving a proper name were generally recalled better than similar propositions involving, e.g., a common noun.

Brown et al. (2008) developed the Computerized Propositional Idea Density Rater (CPIDR) in order to perform an automated analysis of propositional idea density for English texts. Their software approximates propositional idea density by using a combination of part-of-speech information and hand-crafted rules (optimizing their method to match the calculations of (Turner and Greene, 1977). More recently, Cunha et al. (2015) have developed a tool which uses dependency parses to provide more accurate estimates of idea density based on an updated manual for idea density calculation (Chand et al., 2010). This new tool facilitates the use of idea density features in our current task.

4 Learning to Rank

So far we have explored a variety of psycholinguistically-motivated metrics that form the feature-space for our sentences. Every sentence which can be parsed by both ModelBlocks and by the Stanford parser gets mapped into this space consisting of real numbers. Our objective now is to learn a mapping from these feature vectors Φ to the real numbers such that, given two sentences s_i and s_j with the latter more difficult than the former, s_j is mapped to a higher number than s_i.

Machine learning problems wherein items need to be ordered relative to one another but do not necessarily belong to separable classes of data are referred to as learning to rank problems. One classic approach to the problem involves reframing ranking as binary classification of pairs of items. We explore this approach in Section 4.1 before explaining its implementation in an averaged perceptron learner in Section 4.2. In Section 4.3 we provide some brief background on SVM^{rank} , which automates the process of generating pairwise comparisons in its support vector machine approach to the problem.

4.1 Ranking as classification

Standard classification approaches do not work for our ranking problem. While our sentences have only two or three labels (i.e. ranks), we cannot treat these as mutually exclusive classes: a sentence A with rank 1 relative to sentence B with rank 2 may remain difficult (i.e. rank 2) with respect to a third sentence C (with relative rank 1). This means that we cannot simply treat rank labels as labels in a multi-class classification task.[1] Vajjala & Meurers (2014) confirmed this reasoning on the PWKP corpus (Zhu et al., 2010), finding that treating English and Simple English

[1]See Figure 2.1 in §2.3 for an example of such a distribution of difficulty levels in a corpus of sentences from English and Simple English Wikipedia.

as non-overlapping classes resulted in a classification accuracy of only 66%. Fortunately, this sort of relative ranking *can* be reformulated as a binary classification task, classifying pairs of items.

Suppose we have a set of sentences $S = s_1, s_2, \ldots, s_n$. Let $\Phi(s_i) \in \mathbb{R}^d$ represent the d-dimensional feature vector for each sentence s_i. We would like to learn a scoring function σ such that $\sigma(s_i) < \sigma(s_j)$ if and only if $rank(s_i) < rank(s_j)$.

Consider a simple linear model for σ:

$$\sigma(x_i) = w \cdot \Phi(x_i) \tag{4.1}$$

with $w \in \mathbb{R}^d$. Imposing the constraint that $\sigma(s_i) < \sigma(s_j)$ iff. $rank(s_i) < rank(s_j)$, we have

$$\begin{aligned}
\sigma(s_i) &\leq \sigma(s_j) && (4.2)\\
w \cdot \Phi(s_i) &\leq w \cdot \Phi(s_j) && (4.3)\\
0 &\leq w \cdot \Phi(s_j) - w \cdot \Phi(s_i) && (4.4)\\
0 &\leq w \cdot (\Phi(s_j) - \Phi(s_i)) && (4.5)
\end{aligned}$$

where we recognize $\Phi(s_j) - \Phi(s_i)$ as the vector of *difference features* for items s_j and s_i. Referring to s_j as the COMPLEX item and s_i as the SIMPLE item, we see that learning the weight vector w which correctly ranks pairs of items in S is equivalent to learning the weight vector which correctly classifies difference feature vectors as COMPLEX-FIRST or SIMPLE-FIRST, according to which item is the *minuend*. Indeed, this is borne out by Herbrich et al. (1999), who show that such transformations of the ranking problem to a pairwise classification problem are equivalent.

This formulation of the classification problem fits directly into the perceptron algorithm and its extension, the averaged perceptron algorithm. While any algorithm for binary classification could be used, the ability to re-use the weight vector learnt in this manner for scoring items in the original dataset is computationally convenient. When applying the model to new data, we do not have to explicitly compute difference features for every possible pair of items we would like to rank, but rather we can compute the score and use the normal ordering on the real numbers to order

the items by their score. That is, the weight vector w is the same for the scoring function for *any* sentence, not just for the vectors of difference features, in principle resulting in a model which can be used off-the-shelf to score sentence complexity.

4.2 Averaged Perceptron Learner

Given data as described in Section 4.1, the perceptron algorithm tries to learn a weight vector which minimizes the number of misclassifications on the training data. This is done by performing online updates during learning every time the model misclassifies an input. Consider the pseudocode in Algorithm 1.

Algorithm 1 Perceptron Algorithm for binary classification with labels $\in \{-1, 1\}$ and feature vector function for items $\Phi(\textsc{Item}) \in \mathbb{R}^d$

1: **procedure** $\textsc{Perceptron}(\textsc{TrainingSet}, N)$
2: $\quad w \leftarrow (0, \ldots, 0) \in \mathbb{R}^d$
3: $\quad$ **for** N iterations **do**
4: $\quad\quad$ **for** $\textsc{Item}, \mathit{label} \in \textsc{TrainingSet}$ **do**
5: $\quad\quad\quad \mathit{prediction} \leftarrow \mathit{sign}(w \cdot \Phi(\textsc{Item}))$ $\quad\triangleright$ the predicted label
6: $\quad\quad\quad$ **if** $\mathit{prediction} \neq \mathit{label}$ **then**
7: $\quad\quad\quad\quad w \leftarrow w + (\mathit{label} - \mathit{prediction}) \cdot \Phi(\textsc{Item})$

For N iterations through the training data, the weight matrix is updated by the difference between the feature vector for the correct class and the feature vector for the incorrect class. In this simple case there is no dependency between the feature vector and the class label, so we have the simplified update step in line 7.

As an online learning algorithm, however, the updates in this algorithm are sensitive to the order of the items in the training set. One result of this is that there is no way to account for the 'lifetime' of a particular set of weights: one weight vector w_i might have to be updated on the very next training example (indeed, it may have been due to noise), while another w_j successfully classifies 20% of the training data correctly before being updated. This problem is addressed by the voted perceptron algorithm of Freund & Schapire (1999), which is explained in (Collins, 2002). Collins proposed the voted perceptron algorithm for the re-ranking the output of a named-entity extraction system and found that it outperformed the maximum entropy baseline while being very efficient to train.

Unfortunately, the voted perceptron results in a less efficient testing algorithm, since a given feature vector must be evaluated against every weight vector, weighted by the number of training items the weight vector survived, for a vote to be taken. The voted perceptron can be approximated by the averaged perceptron algorithm (Algorithm 2) with guarantees for the resulting performance. The averaged perceptron instead requires that only an average weight vector is learnt, which reduces the complexity of the testing algorithm to the same as the plain perceptron algorithm.

Algorithm 2 Averaged Perceptron Algorithm for binary classification with labels $\in \{-1, 1\}$ and feature vector function for items $\Phi(\textsc{Item}) \in \mathbb{R}^d$

```
1: procedure AveragedPerceptron(TrainingSet, N)
2:     w ← (0, ..., 0) ∈ R^d
3:     w_avg ← (0, ..., 0) ∈ R^d                ▷ initialize the average weight vector
4:     for N iterations do
5:         for Item, label ∈ TrainingSet do
6:             prediction ← sign(w · Φ(Item))
7:             if prediction ≠ label then
8:                 w ← w + (label − prediction) · Φ(Item)
               w_avg = w_avg + w                ▷ update the average weight vector
9:     w_avg ← w_avg / (|TrainingSet| · N)      ▷ take the average
10:    return w_avg                             ▷ returns the averaged weight vector
```

4.3 SVM^{rank}

In this thesis we are focusing on the application of ranking to the measurement of text complexity, but ranking is an important problem in machine learning more generally, with applications in information retrieval, named entity and part-of-speech tagging, parsing, and generation. One well-performing implementation is SVM^{rank} (Joachims, 2006), developed in the context of re-ranking search results in information retrieval (Joachims, 2002).

Support Vector Machines (SVMs), also known as Support Vector Networks, were introduced by Cortes & Vapnik (1995) as a model related to simple feed-forward neural networks (the linear forms of which are perceptrons). This model was transformed into a *preference ranking* problem in (Joachims, 2002).

One of the nice properties of this system is that it assumes only partial knowledge

of the ranking order to be learned. That is, we do not necessarily have fixed ranks 1, 2, 3, ... such that all items of rank 3 should be more highly ranked than all items of rank 2. Rather, within each block of training data[2] the relative ranking of the items must hold. This is a good fit for the model of complexity we want to achieve: we only have individual pairs (or triples) of sentences expressing the same content at different (relative) levels of complexity, but from these items we wish to induce a general scale for complexity. This is analogous to learning a single weight vector (or a single set of support vectors) which can be applied across training samples to generate locally-relevant rankings.

4.4 Relevance to this thesis

Linear models have been shown to perform well in readability tasks over the years and the theoretical formulation of our problem suggests that using a perceptron learner for our models is reasonable. We also examine SVM^{rank} as SVMs in general have been used in a variety of text classification and ranking tasks, including by Vajjala & Meurers (2014).

[2] In Chapter 5, for example, each triple of sentences from the OSE corpus.

5 Methodology

We use two corpora of aligned sentences at different levels of difficulty in two different machine learning paradigms. These corpora differ in terms of the source of their notion of complexity and their size. For both of the machine learning approaches—the averaged perceptron model and SVM^{rank}—we use ten-fold cross-validation to evaluate performance. The models are built using different sets of features based on the psycholinguistic metrics described in Chapter 3.

5.1 Corpora

We use two corpora with different properties for this study. The first is a set of aligned sentences from English and Simple English Wikipedia from Hwang et al. (2015, ESEW). The second is a set of professionally edited texts for learners of English produced by a British newspaper (Macmillan Publishers, 2015), aligned by Vajjala (2015).

5.1.1 ESEW Corpus

The ESEW corpus is large, consisting of more than 150k aligned pairs of sentences in the 'good' subset of the corpus—the portion of the corpus with the highest similarity scores between the English and the Simple English sentence in each pair. For our purposes, we treat the labels on these pairs as a gold standard for complexity. That is, for each sentence pair, we assume the sentence from English Wikipedia is in fact more complex than the sentence from Simple English Wikipedia. While this is the largest available collection of aligned sentence pairs at two different reading levels, the reading levels themselves are not well-defined. Since Wikipedia is a collaborative

Rank	Sentence
2	Gingerbread was brought to Europe in 992 by the Armenian monk Gregory of Nicopolis -LRB- Gregory Makar -RRB- -LRB- Grégoire de Nicopolis -RRB- .
1	Armenian monk Gregory of Nicopolis -LRB- Gregory Makar -RRB- -LRB- Grégoire de Nicopolis -RRB- brought ginger bread to Europe in 992 .

Table 5.1: Example sentences from English (2) and Simple (1) English Wikipedia.

encyclopedia that anyone can edit, in many cases without even registering, it is likely to reflect many different interpretations of 'Simple' English.[1]

Furthermore, there is no direct correspondence between articles in the two wikis. Hwang et al. (2015) follow earlier work in aligning articles from English to Simple English Wikipedia based on shared titles, but the two articles may have been independently written: in some cases an author will create a Simple English article by copying and then editing an English article, but this is not always the case. This complicates the problem of aligning sentences between the two articles, although Hwang et al. (2015) improved on earlier alignments (Zhu et al., 2010)— which relied on sentence-level TF*IDF measures to align sentences—by using a greedy search over a Wiktionary-derived semantic similarity score.

After filtering the corpus to remove 'sentences' longer than 160 words[2] and exact matches (pairs of sentences which were identical in both corpora), we were left with 117 169 sentence pairs, of which we set aside 10% as development data.

[1] Note that Wikipedia does provide guidelines for how to edit the Simple English wiki, emphasizing the BASIC English guidelines (Wikipedia, 2015a) and saying "Articles do not have to be short to be simple; expand articles, add details, but use basic vocabulary" (Wikipedia, 2015b), but there is no guarantee that any given author or editor applies these guidelines.

[2] As you might have gathered, Wikipedia is a somewhat noisy data source. One kind of 'sentence' we need to exclude is that of the form *Top grossing films include: Captain Jack and the Magical Pony, 2001; The Neverending Thesis, 2015; Kapitel Punnish Mints, 1987; . . . (more film titles and years, up to 160 words) . . . and Judgemental Fox is NOT Impressed, 1991.* While a list of this form might obey orthographic conventions related to sentence structure, it is surely not what we should consider a sentence.

Rank	Sentence
3	It is a work-hard, play-hard ethic that many of the world's billionaires might subscribe to but it would be a huge change for most workers and their employers.
2	It is a 'work-hard, play-hard' way of thinking that many of the world's billionaires might agree with but it would be a huge change for most workers and their employers.
1	Many of the world's billionaires might agree with this way of thinking but it would be a very big change for most workers and their employers.

Table 5.2: Example sentences from One Stop English, at levels advanced (3), intermediate (2), and elementary (1).

5.1.2 One Stop English Corpus

The One Stop English corpus (OSE) is about 100 times smaller than the ESEW corpus, consisting of 1577 triples of sentences. onestopenglish consists of news stories edited to three different reading levels: elementary, intermediate, and advanced. To construct the corpus, Vajjala (2015) extracted the text of each triple of articles and used TF*IDF and cosine similarity to align sentences between these articles. A second portion of this corpus, consisting of those sentences which did not appear in all three versions of an article, but only in two versions of the article, is also available.[3]

One Stop English does not appear to provide information about the way their texts are assigned to a particular reading level, but these texts are at least professionally drafted as pedagogical resources for language learners. This targeted authorship and editing should result in a more coherent notion of text complexity exemplified in the corpus, though it is difficult to be certain without more information from the publisher.

[3]The collection of sentence pairs is referred to as OSE2 by Vajjala (2015). Since only the collection of triples, OSE3, is used in this thesis, I refer to it simply as OSE.

5.2 Feature Extraction

We extract a total of 22 features from both corpora using two parsers. The `ModelBlocks` parser (van Schijndel and Schuler, 2012) provides features based on surprisal and embedding depth while the Stanford parser[4] provides dependencies which are used to calculate integration cost as well as idea density.

5.2.1 Baseline Features

We use the two features most commonly used in document-level readability formulae as our baseline features:

- sentence length (sentLength) measured in words; and
- average word length (avgWordLength) measured in characters per word.

Here we are referring to 'words' as non-whitespace sequences of characters separated by whitespace following tokenization.

We also considered using the maximum word length of a sentence, but there is too much arbitrary variability to make this a meaningful feature. While simpler texts may avoid using longer words, some long words are unavoidable given a particular topic and would therefore be likely to appear in both easier and harder sentences on the same topic.

Note that when we add interaction terms to our models, the interaction term for this feature set is equal to the sentence length in characters.

5.2.2 Surprisal Features

Based on UID and earlier work on surprisal (see §3.1), we take the following features from `ModelBlocks`' complexity output:

[4] `http://nlp.stanford.edu/software/lex-parser.shtml`

- lexical surprisal (avgLexSrp and maxLexSrp); and
- syntactic surprisal (avgSynSrp and maxSynSrp).

We use the average measures of surprisal (avgLexSrp and avgSynSrp) as idealized measures of the channel capacity required to process a given sentence. While this underestimates the amount of cognitive resources that is actually required, the measure is at least internally consistent: a sentence with higher average surprisal overall likely requires a higher channel capacity as well.

On the other hand, the maximum surprisal features (maxLexSrp and maxSynSrp) are a measure of the maximum demand on cognitive resources during processing of a given sentence. While the probability of encountering a surprising word will in general be higher in longer sentences, we do not believe this relationship is strictly linear and so we do not try to normalize these features by sentence length.

5.2.3 Embedding Depth Features

`ModelBlocks` also provides measures of embedding depth (see §3.2), which we extract:

- embedding depth (avgEmbeddingDepth and maxEmbeddingDepth); and
- embedding difference (avgEmbeddingDiff and maxEmbeddingDiff).

The average embedding depth can be thought of as a general measure of memory load throughout the processing of the sentence. More difficult sentences will have more embedded constructions, increasing the average embedding depth. While the average embedding depth is a reasonable measure of the average memory load, it is possible that peaks in memory load which do not strongly affect the average embedding depth would still cause processing difficulty. For this reason we also include the maximum embedding depth feature.

Embedding difference, on the other hand, is a measure of how many times a reader (or listener) has to push a construction to or pop an item from their memory store. When reading a sentence with only a single level of embedding, it the embedding

difference will be zero throughout the sentence. While the maximum embedding difference will only rarely differ by much from 1, the fact that these features are calculated over the beam of possible parses in `ModelBlocks` means that the feature is not strictly discrete. Therefore it is possible that a sentence whose structure invites more plausible analyses with a larger difference in embedding depth—and therefore might be more difficult—would be identifiable based on this feature.

5.2.4 Integration Cost Features

The Stanford parser provides dependency parses which can be analysed using `icy-parses` to produce the following integration cost (see 3.3) features:

- total integration cost (totIntegrationCost)
- average integration cost (avgIntegrationCost)
- maximum integration cost (maxIntegrationCost)

Average integration cost is another kind of average memory load measure, while maximum integration cost tracks the most difficult integration a subject encounters in a given sentence. The total integration cost represents an unnormalized measure of the difficulty due to integration throughout the entire sentence.

Again, the maximum and total measures for this set of features are likely to be related to the length of the sentence: in a longer sentence, longer distance dependencies are possible. The avgIntegrationCost measure captures a normalized verion of the totIntegrationCost, but we make no attempt to normalize the maximum integration cost, again assuming that the relationship between this feature and sentence length is not simply linear.

5.2.5 Idea Density Features

The *IDD3* library for Python[5] operates on Stanford dependency parses to produce idea density (see §3.4) features:

- idea density (totIdeaDensity);
- predications (predIdeaDensity);
- modifications (modIdeaDensity); and
- connections (connIdeaDensity).

Cunha et al. (2015) developed the `IDD3` library for the automatic extraction of idea density. Previous work by Brown et al. (2008) approximated idea density using a rule-based part-of-speech tagger and standardized its results against the manual provided by Turner & Greene (1977). In addition to using a more linguistically informative input format, Cunha et al. evaluated their performance against an updated diagnostic (Chand et al., 2010), addressing some of the weaknesses of the *CPIDR* system of (Brown et al., 2008).

This system also provides some differentiation in the kind of idea being expressed. Simple predications and modifications are fairly straightforward, while connections are more complex. In terms of the 'gist' of a passage, predications are most central, followed by modifications, and finally by connections.

Since the crucial measure here is idea density rather than a count of the number of ideas being expressed in a given sentence[6], we divide the count of each type of idea by the number of words in the sentence. Note that totIdeaDensity is simply the sum of the other three forms of idea density.

[5] `https://github.com/andrecunha/idd3`; our fork available from `https://github.com/dmhowcroft/idd3`

[6] This is especially true considering that good simplifications should preserve semantic content.

5.3 Models

We train an Averaged Perceptron model in Python, treating the ranking problem as a classification problem based on difference features. To address the ranking problem directly, we use SVM^{rank} (Joachims, 2006). Both approaches are trained on the following feature sets:

- Baseline – sentenceLength and avgWordLength
- Surprisal – all surprisal features
- Embedding – all embedding depth and difference features
- IntegrationCost – all integration cost features
- IdeaDensity – all idea density features
- Psycholinguistic – all psycholinguistically-motivated features; i.e. Surprisal+Embedding+IntegrationCost+IdeaDensity
- ModelBlocks – psycholinguistic features derived from `ModelBlocks` i.e. Surprisal+Embedding
- Stanford – psycholinguistic features derived from the Stanford parser i.e. IntegrationCost+IdeaDensity
- Baseline+ModelBlocks – Baseline features plus those extracted from `ModelBlocks`
- Baseline+Stanford – Baseline features plus those extracted from the Stanford parser
- FullModel

The two models grouping features together based on the parser they are derived from are included for practical considerations. Since the `ModelBlocks` parser is substantially slower than the Stanford parser, we would rather run only the Stanford parser if a particular application requires especially fast sentence rankings. In order

to determine if the accuracy gain from running `ModelBlocks` justifies the delay in sentence processing, we need to evaluate models based on these feature subsets as well.

5.3.1 Averaged Perceptron

For the averaged perceptron model we use an implementation in the Python programming language (see Appendix B). We examine two different classification tasks with the averaged perceptron learner. First, we take pairs of sentences at different difficulty levels and compute a difference vector from their feature vectors. Half of these difference vectors are then multiplied by negative one, providing a second class for the classification task. For example, with the ESEW data, we subtract the vector representing the Simple English sentence from the vector representing the English sentence. This difference vector is now an example of the EnglishFirst (or ComplexFirst) class. Trading half of these vectors for their additive inverses provides training examples for the SimpleFirst class. In this way we train a model which identifies which of a pair of sentences is more complex.

The second approach includes z-score normalization of the difference features. While such normalization can improve performance, it diminishes the transferability of these results back to the single sentence (i.e. non-difference) feature space.

5.3.1.1 Further Data Splits

Given the OSE reading levels Elementary, Intermediate, and Advanced, we have three possible datasets generated from difference features. These are obtained by subtracting the features of Elementary sentences from those of their Intermediate counterparts (Int-Elem), subtracting Intermediate from Advanced (Adv-Int), and subtracting Elementary from Advanced (Adv-Elem).

Of these subsets, Adv-Int and Int-Elem represent finer-grained differences: are we able to tell the difference between two levels which, in theory, are less strongly differentiated from one another. The combined dataset consisting of these two subsets is called the OSE_{Close} dataset. The OSE_{Close} dataset stands in contrast to the $OSE_{Adv-Elem}$ dataset, which represents a coarser distinction in readability levels.

5.3.2 SVM^{rank}

We use the default configuration for SVM^{rank} throughout our analyses, with further details reported in Appendix C. Here there is no need to calculate different features as SVM^{rank} operates directly on the relatively-ranked input pairs or triples. There is a single parameter for SVM^{rank} , however, which represents the "trade-off between training error and the margin"[7]. As the choice for this parameter can affect the accuracy of the learnt model, we perform grid search over possible values for this parameter in order to select the best model.

[7]Quoted from the documentation available at `http://svmlight.joachims.org/svm_rank.html`

6 Features in the Corpora

Before testing our hypotheses, we examine the distributions of each feature in each corpus. This provides a description of the properties of the corpora with respect to the features under consideration and also a qualitative understanding of the features themselves.

Each section below includes a box-and-whiskers plot of the distribution for each feature in each corpus slice, where the slices are: Simple English Wikipedia, English Wikipedia, OSE Elementary, OSE Intermediate, and OSE Advanced.

6.1 Baseline Features

Both sentence length and average word length features exhibit increasing means with increasing difficulty level, and maximum word length also shows this difference except between the Elementary and Advanced levels of the OSE corpus. That said, the distributions across the levels overlap considerably. The interquartile range for average word length is consistent across corpora. See Figures 6.1 and 6.2.

6.2 Surprisal Features

All of the corpus-mean values for surprisal features show the same pattern: higher values for more difficult corpora. All of the averaged features have similarly large interquartile ranges and overlapping distributions. For the maximum features, lexical surprisal (and therefore total surprisal) shows a greater variance in the OSE corpora than in the ESEW corpora. See Figures 6.3, 6.4, 6.5, and 6.6.

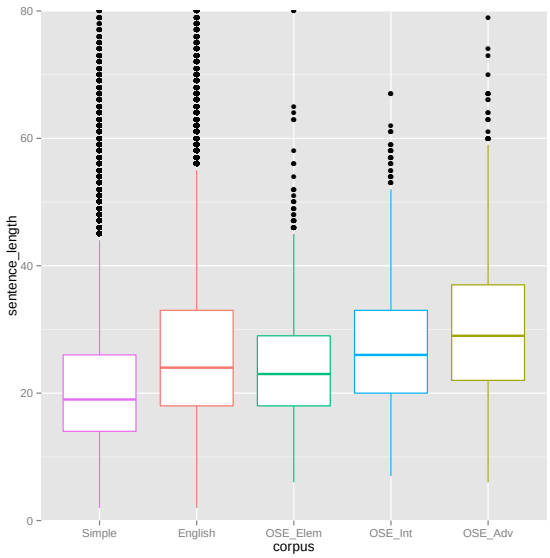

Figure 6.1: Sentence length in words across our corpora

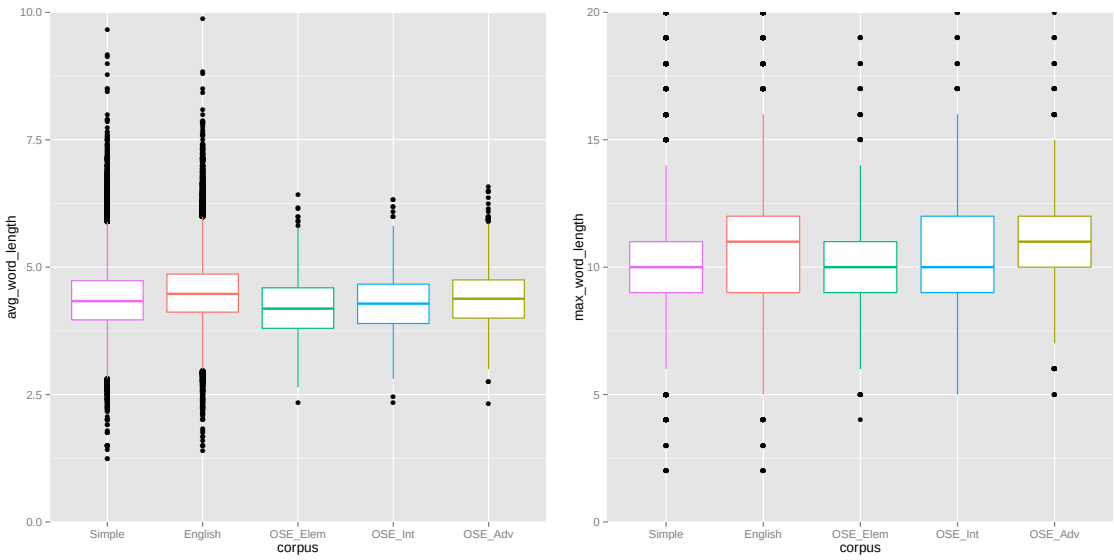

Figure 6.2: Average and maximum word lengths in characters across our corpora

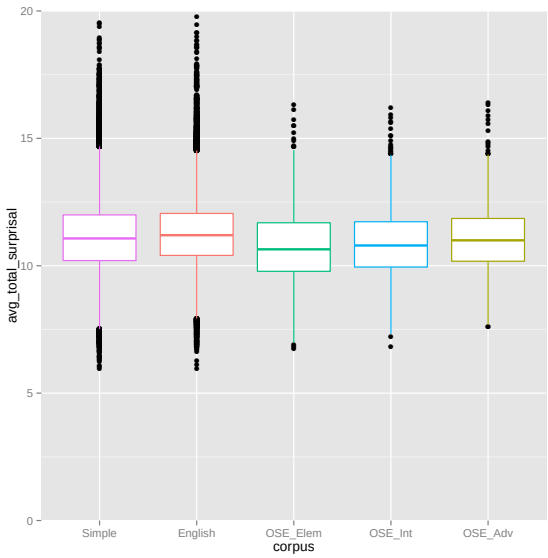

Figure 6.3: Average total surprisal across our corpora

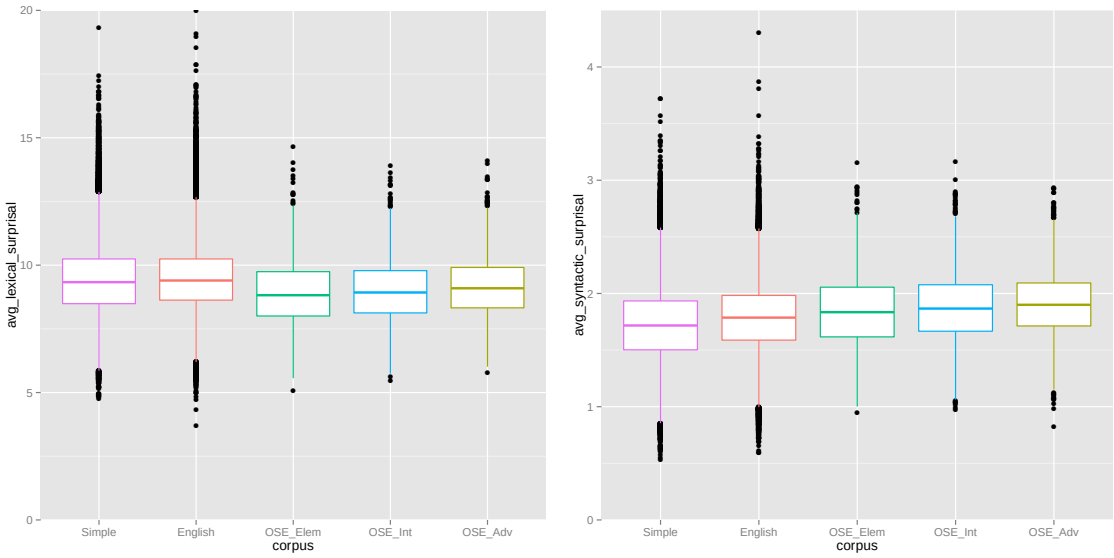

Figure 6.4: Average lexical and syntactic surprisal across our corpora

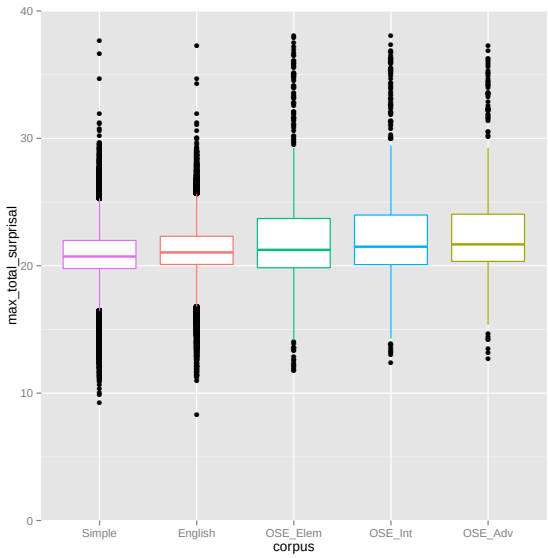

Figure 6.5: Maximum total surprisal across our corpora

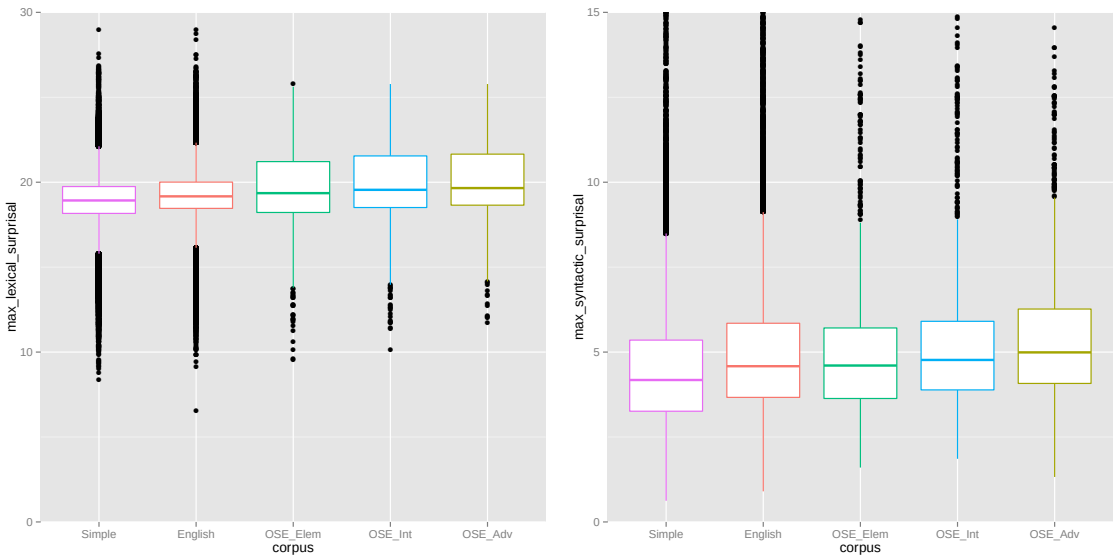

Figure 6.6: Maximum lexical and syntactic surprisal across our corpora

6.3 Embedding Depth and Difference

Average and maximum embedding depth both pattern according to our intuitions: increasing with difficulty level. Average embedding *difference*, on the other hand, runs counter to our intuition, suggesting that simple sentences are more likely to include multiple shifts in embedding depth than more difficult sentences. As predicted, the maximum embedding depth is almost always 1, suggesting that it is a minimally useful feature that might reasonably be binarized in future work. See Figures 6.7 and 6.8.

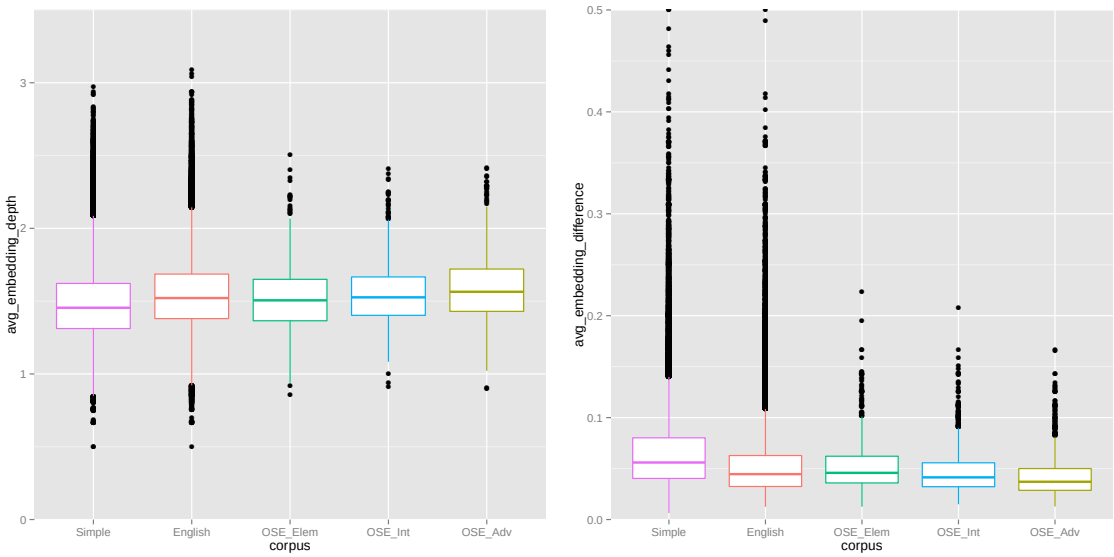

Figure 6.7: Average embedding depth and difference across our corpora

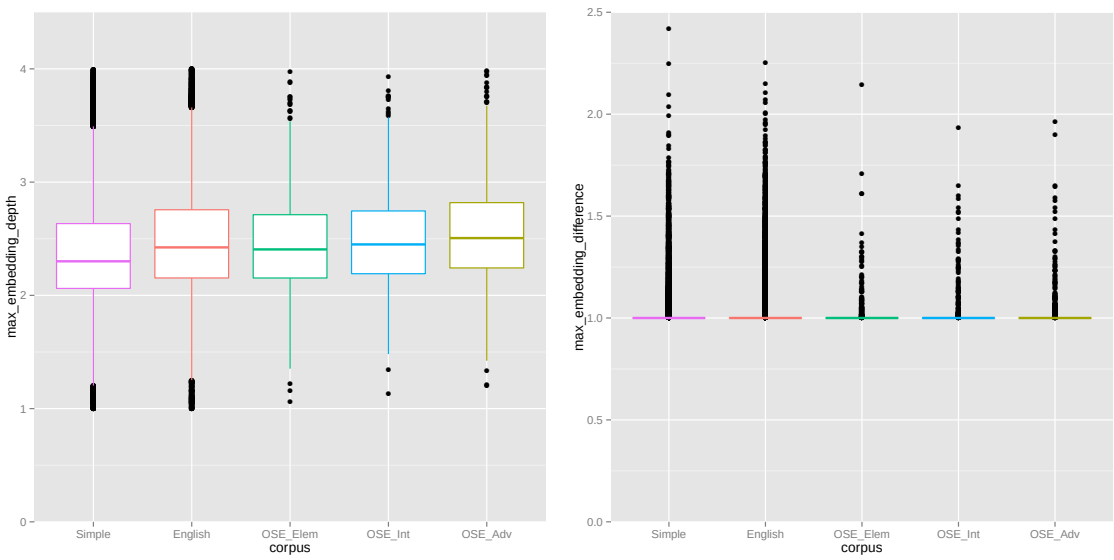

Figure 6.8: Maximum embedding depth and difference across our corpora

6.4 Integration Cost

Total integration cost increases as text difficulty increases, as expected due to longer sentences in more difficult texts. Average integration cost, which is basically a sentence-length-normalized version of total integration cost, also exhibits this property, though the differences in the means and the distributions is very slight indeed. The means for maximum integration cost appear to be almost identical within-corpus (i.e. the means for Simple English and English wikipedia appear to be the same and the means across the OSE subcorpora also appear to be the same). That said, the range of values is higher for more difficult texts, suggesting at least some potential to discriminate between the levels with this feature. See Figures 6.9 and 6.10.

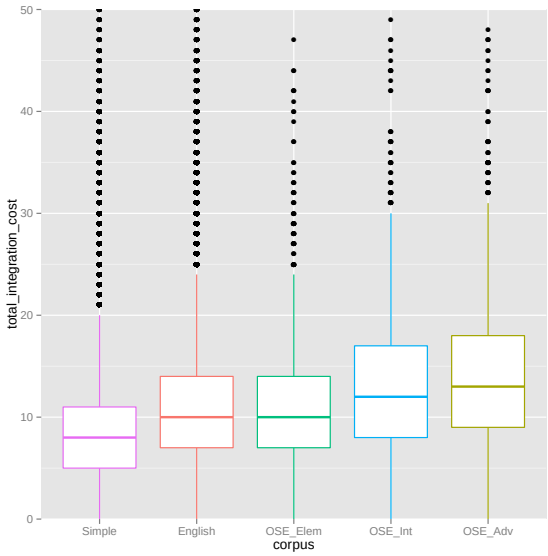

Figure 6.9: Total integration cost across our corpora

6.5 Idea Density

Counter-intuitively, idea density appears to decline as text difficulty increases, an effect largely driven by predications (the top-right graph of `prop_idea_density`), which tend to be expressed as verbs. In contrast, modifiers appear fairly stable across the corpora, and connection density is almost always zero (the bottom two graphs in Figure 6.11). Simpler sentences do tend to be shorter, and there is a lower bound on the number of verbs that can be in a sentence, partially explaining this trend. Further, it has been observed that more advanced texts tend to be more

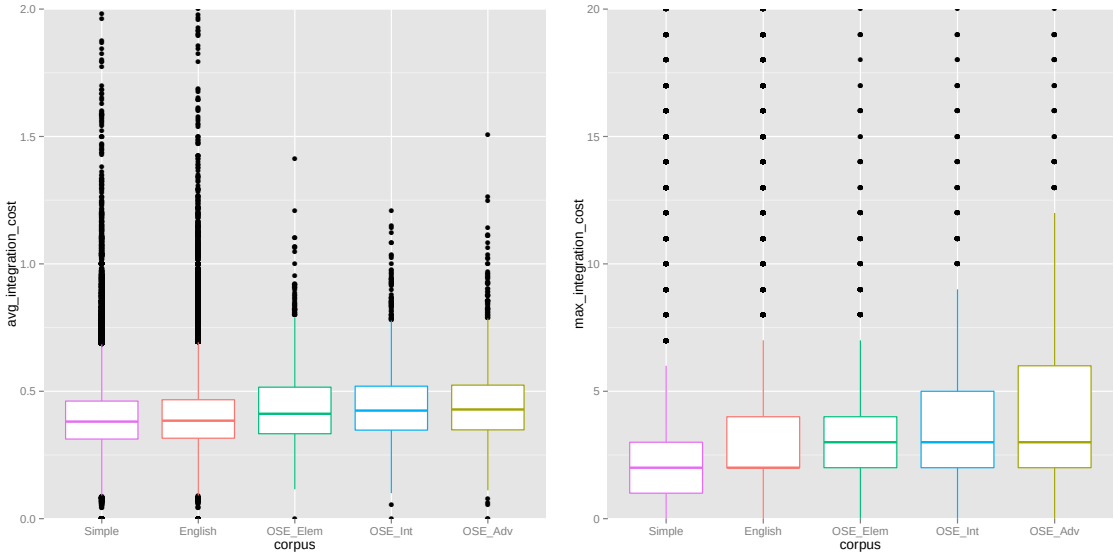

Figure 6.10: Average and maximum integration cost across our corpora

reliant on nominalizations, which are less likely to be counted as contributing to the idea density of a text in this formulation. See Figure 6.11.

6.6 Remarks

The features under examination generally behave as expected across the difficulty levels in our corpora. However, the choice to approach the problem as one of relative ranking is reinforced by the amount of overlap in the distributions. The counter-intuitive behavior of idea density also invites further investigation, as it remains unclear exactly what this measure is really capturing about sentence processing.[1]

[1]For further criticism of idea density as a measure in psychological studies, see King (2012).

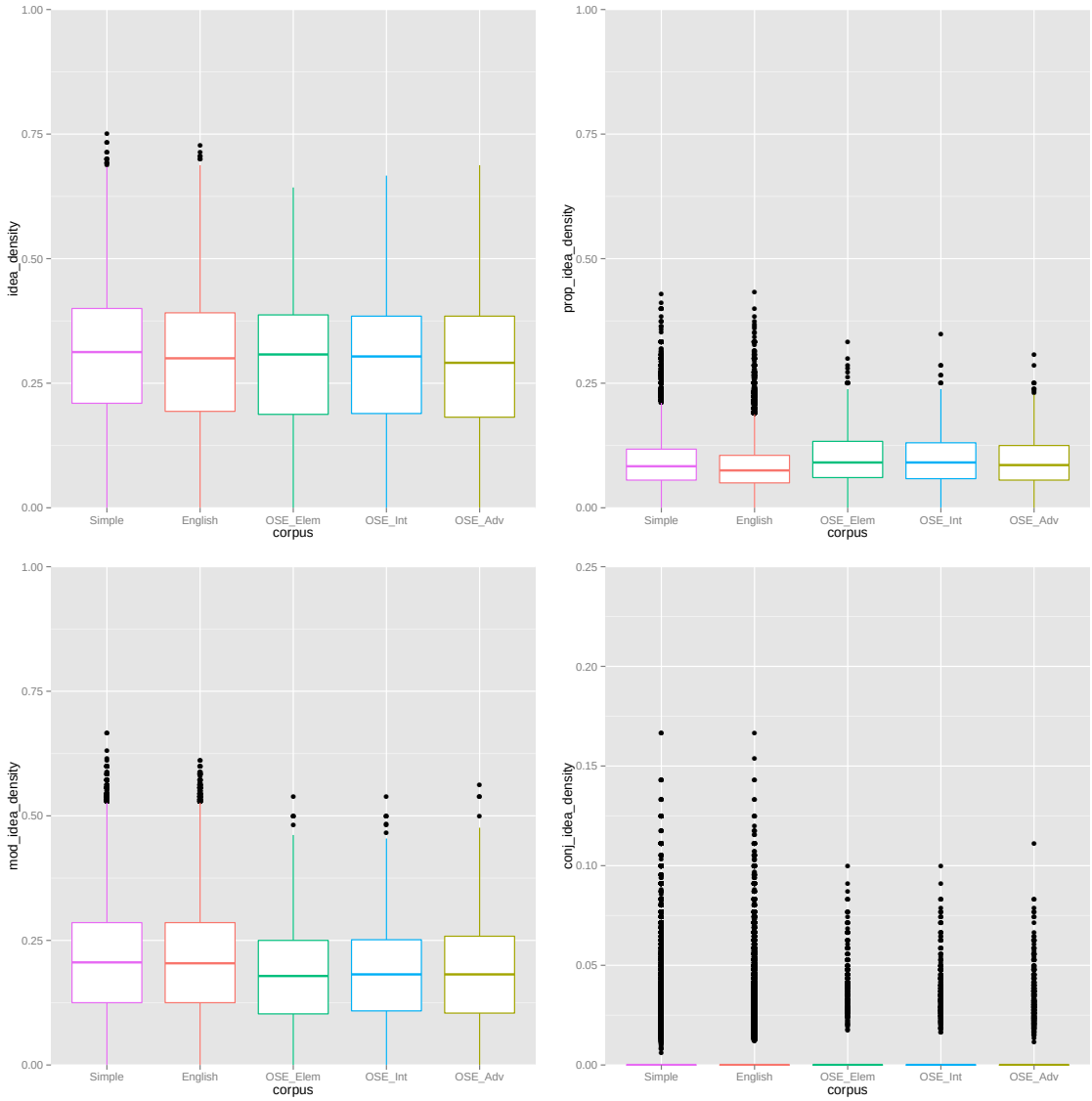

Figure 6.11: Idea density measures across our corpora

7 Analysis and Results

7.1 Predictions

In this thesis we are examining the utility of features extracted from psycholinguistic models of sentence processing for predicting sentence difficulty as determined by human annotations. While average sentence length and average word length provide reasonable results when levelling whole documents, we expect that the theoretically-motivated measures incorporated into our features will improve model performance over this baseline. This yields the following main hypothesis:

Main Hypothesis

Psycholinguistic features will improve performance over the Baseline

Specifically, we expect Baseline+ModelBlocks, Baseline+Stanford, and FullModel to have significantly higher accuracies than the Baseline model. In our significance testing, we will also compare the performance of these models to each other, with appropriate corrections for multiple comparisons.

In addition to explicitly testing this hypothesis, we run models using subsets of the features to collect descriptive statistics comparing the performance of the features rooted in different psycholinguistic theories of sentence processing. This is our feature comparison analysis.

Exploratory Analysis: Feature Comparison

Examine which types of psycholinguistic features provide the greatest gain in performance

A secondary hypothesis is that these psycholinguistic features will be most helpful in finer-grained distinctions, as opposed to coarser distinctions. To examine this hypothesis we will compare the performance of the Averaged Perceptron model on those pairs of sentences from the *Advanced* and *Elementary* portions of the OSE corpus ($OSE_{Adv-Elem}$) to its performance on $OSE_{Close} = OSE_{Adv-Int} \cup OSE_{Int-Elem}$. For this comparison we are not examining the same underlying data, so we do not perform any significance testing between the subcorpora.

Secondary Hypothesis

Psycholinguistic features are more important for fine-grained distinctions between sentences than for coarse distinctions.

7.2 Statistical Analyses

Each model is evaluated using ten-fold cross-validation, effectively creating ten versions of each model, one for each fold of the dataset. Combining the predictions of these ten submodels yields a prediction for each relatively-ranked sentence pair in the corpora.[1] Since each model is making predictions on the same set of samples, we can use McNemar's test (McNemar, 1947) for paired samples.[2]

To correct for the multiple comparisons we are making in evaluating these different models, we use the fairly conservative Bonferroni correction. For the purposes of evaluating our main hypothesis, we consider only the paired comparisons

[1] See Chapter 4 for details on how the models make their predictions and how sentences are grouped into pairs for training and testing.

[2] Note that when the discrepancy in model predictions is too small one must use the sign test instead, but our check for this criterion identified no model pairs which needed to be evaluated in this way.

between the 4 relevant models: BASELINE, BASELINE+MODELBLOCKS , BASELINE+STANFORD , and FULLMODEL. For the purposes of our feature comparison, we pretend that we are testing a particular hypothesis on all of the model differences, which reinforces the already conservative Bonferroni correction. This increases our confidence that the differences reported to be significant are real differences, even for our exploratory analysis.

Note that Appendix D includes more detailed information about these analyses and their outcomes, including uncorrected p-values and effect sizes.

7.3 Analysis of Perceptron Results

7.3.1 Evaluating the Main Hypothesis

Table 7.1 shows the accuracy of the four models relevant to our main hypothesis when operating on raw difference features. The first two columns represent the corpora of interest to the main hypothesis. The full model, including all of our psycholinguistic features, performs significantly better than the BASELINE. For the OSE corpus, the difference between BASELINE+MODELBLOCKS and the FULLMODEL is not significant, but both of these models are significantly better than the BASELINE for both corpora. The BASELINE+STANFORD model has a significant improvement of one percentage point over the BASELINE for the $ESEW$ corpus but the smaller improvement on the OSE corpus is not significant.

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
BASELINE	71.24	‡75.10	+82.11	‡71.36
BASELINE+MODELBLOCKS	72.58	†**77.94**	+**85.29**	†**74.61**
BASELINE+STANFORD	72.39	‡75.50	82.87	‡71.85
FULLMODEL	**73.22**	†77.24	84.38	†74.23

Table 7.1: Averaged Perceptron performance on difference features using 10-fold cross-validation. All differences are significant in the $ESEW$ column. In OSE_{All} and OSE_{Close} the difference between items marked with the same symbol are not different, but all others are. For $OSE_{Adv-Elem}$ only the difference between the BASELINE and the BASELINE+MODELBLOCKS systems is significant. (This is denoted with the + sign.)

Table 7.2 shows the accuracy of the same features over the same corpora when the features have been z-score normalized. For $ESEW$ the FullModel is again significantly better, though the improvement is now slightly less than 0.4 percentage points (as opposed to the $\approx$ 2 percentage points for non-normalized features). For OSE, all of the models achieve the same performance of $\approx$ 78.5, except for Baseline+Stanford , which performs (significantly) about 2 percentage points worse.

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Baseline	73.37	=78.14	=84.69	74.64
Baseline+ModelBlocks	72.91	=**78.87**	=**85.29**	75.81
Baseline+Stanford	−**73.81**	76.93	=83.63	+73.97
FullModel	−73.76	=78.32	=84.53	+**76.04**

Table 7.2: Averaged Perceptron results for z-score normalized difference features. For $ESEW$, all differences except the one marked by − are significant. For OSE_{All}, only Baseline+Stanford is significantly different from the other models. For $OSE_{Adv-Elem}$, no differences are significant. For OSE_{Close}, only the difference between Baseline+Stanford and FullModel are significant. Values within a column marked with = are not significantly different from each other.

Table 7.3 shows the accuracy over the same corpora (with non-normalized difference features) when interaction terms for all are added. Again, all differences are significant for $ESEW$, even though the FullModel gets exactly the same accuracy as the Baseline. The best model for both $ESEW$ and OSE is the Baseline+ModelBlocks model, which also outperforms the FullModel in this case.

Table 7.4 shows the accuracy over the same corpora (with z-score normalized difference features) when we include interaction terms. For the $ESEW$ corpus, most differences are again significant and we find that the FullModel outperforms the Baseline by more than one percentage point. Baseline+ModelBlocks achieves the highest accuracy of any of our models on the OSE dataset, though the difference between this model's performance and the Baseline is not significant. See, however, the discussion in §7.5 of an error in the implementation of z-score normalization which may partially explain the much-improved performance of the Baseline models in the evaluations involving normalized data.

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Baseline	72.20	−76.71	=84.98	72.68
Baseline+ModelBlocks	**72.71**	**79.62**	**=86.87**	**76.08**
Baseline+Stanford	70.99	75.40	82.65	71.70
FullModel	72.20	−78.57	=86.42	75.10

Table 7.3: Averaged Perceptron performance on difference features with added interaction terms using 10-fold cross-validation. All models are significantly different on the $ESEW$ dataset, including the two models with identical performance. The only difference that is not significant for OSE_{All} is that between the Baseline and the FullModel. For $OSE_{Adv-Elem}$, only the Baseline+Stanford model is significantly different from the other models. For the OSE_{Close} dataset, each model is significantly different only from the next model in the table. That is, Baseline differs from Baseline+ModelBlocks which differs from Baseline+Stanford which differs from FullModel, but no other differences are significant.

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Baseline	−73.79	78.74	†85.51	75.06
Baseline+ModelBlocks	−73.71	**79.90**	‡**87.93**	+**76.23**
Baseline+Stanford	74.47	77.04	†84.98	+73.89
FullModel	**75.09**	78.19	‡**87.93**	75.13

Table 7.4: Averaged Perceptron performance on z-score normalized difference features with added interaction terms using 10-fold cross-validation. For $ESEW$, the only non-significant difference is that between the Baseline and FullModels. For OSE_{All}, Baseline+ModelBlocks is significantly different from Baseline+Stanford and FullModel, but not from the Baseline. Meanwhile, Baseline+Stanford is also significantly different from the Baseline. For $OSE_{Adv-Elem}$, we again get a pattern where Baseline and Baseline+Stanford are statistically indistinguishable but are significantly different from the other pair of models, which are in turn statistically 'equivalent' to each other. Finally, the only significant difference for OSE_{Close} is that between the Baseline+ModelBlocks and the Baseline+Stanford models.

7.3.2 Feature Comparisons

For the purposes of feature comparison, we limit our analysis to the first combination of features and interactions: i.e. we use the results for non-normalized difference

features without interactions.[3] Looking at table 7.5, we see that none of the psycholinguistic feature sets is significantly better than the baseline model on its own or in combination with other psycholinguistic features, though adding these features to the baseline model generally results in a significant improvement.

Across all datasets, IdeaDensity performs quite poorly, especially considering that chance performance for this evaluation is 50%. The features for Embedding and IntegrationCost appear to be the best features, suggesting that both of the parsers have something to contribute as each of these is derived from a different parser.

While the ModelBlocks and Stanford models are not generally distinguishable from one another, it seems that adding the ModelBlocks features to the Baseline features always results in better performance than the adding the Stanford features, at least for this dataset.

7.3.3 Evaluating the Secondary Hypothesis

The results in Table 7.1 show that ModelBlocks features result in a significant improvement of approximately 3 percentage points over the baseline model for both slices of the One Stop English corpus. When we add interaction terms (Table 7.3), there is a larger improvement (3.4 vs. ∼2 percentage points) for the levels which are closer together versus those which are further apart, and the larger difference is significant (unlike the smaller). Unfortunately, the error in our implementation of z-score normalization (further discussed in §7.5) means that we cannot easily use the information in Tables 7.2 or 7.4 to better understand what special advantage, if any, our features have for finer-grained distinctions in reading level. Moreover, the lack of any real difference for the Baseline+ModelBlocks model suggests that our study cannot shed any light on this question.

[3]The accuracies and significances for the other comparisons can be found in Appendix D.

Model	ESEW	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
SURPRISAL	61.48	62.40	$=^{1}$67.25	$=^{1,2}$59.74
EMBEDDING	$=^{1}$65.04	64.93	$=^{1,2}$71.70	$=^{1}$61.82
INTEGRATIONCOST	$=^{2}$63.84	$=^{1}$67.27	$=^{3}$76.46	$=^{3}$62.27
IDEADENSITY	55.33	50.97	54.50	$=^{2}$48.87
PSYCHOLINGUISTIC	69.76	$=^{2}$75.07	$=^{3,4}$80.99	$=^{4}$71.89
MODELBLOCKS	$=^{1}$65.57	$=^{1}$71.05	$=^{2,3}$74.42	$=^{3}$67.36
STANFORD	$=^{2}$64.68	$=^{1}$67.98	$=^{3}$77.51	$=^{3}$62.98
BASELINE	71.24	$=^{2}$75.10	$=^{4}$82.11	$=^{4}$71.36
BASELINE+MODELBLOCKS	72.58	**77.94**	**85.29**	**74.61**
BASELINE+STANFORD	72.39	$=^{2}$75.50	82.87	$=^{4}$71.85
FULLMODEL	**73.22**	77.24	84.38	74.23

Table 7.5: Averaged Perceptron performance on difference features using 10-fold cross-validation. Differences among the four models used for the main hypothesis are not explored here. Their accuracies are reported in order to compare with the feature subsets. All differences except those marked with = are significant. Those marked with = are further annotated with a superscript to indicate equivalence classes. Such classes are valid only within a single column. For example, the difference between EMBEDDING and MODELBLOCKS is not significant for $ESEW$, though it is for OSE_{All} and OSE_{Close}. For OSE_{All}, MODELBLOCKS, INTEGRATIONCOST, and STANFORD are not significantly different from each other. The overall PSYCHOLINGUISTIC model is also not different from the BASELINE or BASELINE+STANFORD models.

7.4 Analysis of SVM^{rank} Results

Our analysis of models trained using SVM^{rank} is limited to a comparison on the development data. Since SVM^{rank} appears to perform comparably to the averaged perceptron models (see Table 7.6), we assume for our current purposes that the results from the averaged perceptron models are basically representative of the performance of these features. Given that both the averaged perceptron approach and our use of SVM^{rank} are learning simple linear models for classification, the similarity seems reasonable.

Model	ESEW	OSE_{All}
Surprisal	61.83	56.69
Embedding	60.85	61.68
Integration	64.90	65.53
IdeaDensity	48.89	47.39
Psycholinguistic	70.28	70.98
Modelblocks	63.64	59.18
Stanford	65.77	66.67
Baseline	72.33	73.47
Baseline+Modelblocks	72.98	74.38
Baseline+Stanford	72.53	73.02
FullModel	73.07	**75.74**
BestPerceptron(Dev)	**73.15**	74.62

Table 7.6: Accuracy **on the development data** after grid-searching for C. Chosen C values reported in Appendix C.

7.5 Comments on norming

Reflecting on the effect of norming in this study, it appears that z-score normalization dramatically improved the effectiveness of the baseline model. The increase in accuracy of 1.5 to 3 percentage points indeed removes much of the gap between the Baseline and the psycholinguistic models. While normalization does generally improve model fit when done appropriately, in this case we believe this is in fact an error. Our normalization process was not adequately separated within the training and testing process, so the mean and standard deviation used to calculate the z-score for each feature was, in fact, calculated over the entire dataset, rather than

only being calculated on the training portion of each of the cross-validation folds. This weakness means that the model performance for evaluations with normed features should be taken with a grain of salt, as they reflect a limited sort of awareness of the test data being encoded in the training data.

7.6 Discussion

The results presented in §7.3.1 provide evidence in favor of our hypothesis: adding psycholinguistically-motivated features to the baseline model resulted in significant improvements over the baseline of ≈ 0.9 percentage points for the FullModel on *ESEW* and ≈ 0.5 percentage points on *OSE*.[4] Tables 7.7 and 7.8 show the differences between each model and the Baseline along with the mean difference for each model. This provides a sort of overview for evaluating the performance of the different models. While the addition of the Stanford features to the Baseline appears to improve performance for the more varied *ESEW* corpus, these features actually appear to hurt performance on the *OSE* corpus. The largest gains over the baseline are for Baseline+ModelBlocks on the *OSE* corpus.

ESEW	No Interactions		With Interactions		
Model	Un-normed	Normed	Un-normed	Normed	Mean
Baseline+ModelBlocks	1.34	−0.46	0.51	N/A	0.35
Baseline+Stanford	1.15	0.44	−1.21	0.68	0.27
FullModel	1.98	0.39	0.00	1.30	0.92

Table 7.7: Difference between each model and the Baseline accuracy, or N/A for non-significant differences. For averaging purposes, N/A is counted as a 0.

OSE	No Interactions		With Interactions		
Model	Un-normed	Normed	Un-normed	Normed	Mean
Baseline+ModelBlocks	2.84	N/A	2.91	N/A	1.41
Baseline+Stanford	N/A	−1.21	−1.31	0.68	−0.46
FullModel	2.14	N/A	N/A	N/A	0.54

Table 7.8: Difference between each model and the Baseline accuracy, or N/A for non-significant differences. For averaging purposes, N/A is counted as a 0.

[4] These averages calculated by summing the differences between the two accuracies when the difference is significant and dividing by the number of evaluations, i.e. 4.

Another observation is that the models' performance on *ESEW* appears to be more similar to their perfomance on OSE_{Close} than to their performance on $OSE_{Adv-Elem}$. All models, the baseline included score better on the $OSE_{Adv-Elem}$ dataset, which corroborates the assertion on the part of the *OSE* developers that there is a greater difference in difficulty between the *Advanced* and the *Elementary* materials than between either of those levels and the *Intermediate* level. This pair of observations has implications for future work on the *ESEW* and other Wikipedia-derived corpora, as this suggests that the difference between an English and a Simple English Wikipedia text is more subtle than one might otherwise assume.

	No Interactions		With Interactions		
Model	Un-normed	Normed	Un-normed	Normed	Mean
ESEW	0.19	−0.90	1.72	−0.76	0.06
OSE	2.44	1.94	4.22	2.86	2.87

Table 7.9: Difference between the Baseline+ModelBlocks and Baseline+Stanford models.

In Table 7.9 we look at the performance of the Baseline+ModelBlocks model as compared to the Baseline+Stanford model. This shows that the Baseline+ModelBlocks model has only a slight, if any, advantage over the Baseline+Stanford model for the *ESEW* corpus, though it is substantially better for the *OSE* corpus.

ESEW	No Interactions		With Interactions		
Model	Un-normed	Normed	Un-normed	Normed	Mean
Baseline+ModelBlocks	0.64	0.87	−0.51	1.38	0.60
Baseline+Stanford	0.83	N/A	1.21	0.62	0.67

Table 7.10: Difference between the FullModel and the Baseline+ModelBlocks and Baseline+Stanford models, or N/A for non-significant differences. For averaging purposes, N/A is counted as a 0.

OSE	No Interactions		With Interactions		
Model	Un-normed	Normed	Un-normed	Normed	Mean
Baseline+ModelBlocks	N/A	N/A	−1.05	1.71	0.17
Baseline+Stanford	1.74	1.39	3.17	N/A	1.58

Table 7.11: Difference between the FullModel and the Baseline+ModelBlocks and Baseline+Stanford models, or N/A for non-significant differences. For averaging purposes, N/A is counted as a 0.

The final comparison between our hypothesis-relevant models in Tables 7.10 and 7.11 shows that the full model is generally better than the BASELINE+STANFORD model, at least for the *OSE* corpus, though it is at times indistinguishable from the BASELINE+MODELBLOCKS model.

7.7 Comparison to Vajjala (2015)

We have already reported Vajjala's (2015) performance on the PWKP corpus of Zhu et al. (2010) in §2.3. In ten-fold cross-validation Vajjala achieved an 82.7% accuracy, which decreased to 79.7% when evaluated on separate held-out test data. While these scores are on a different corpus and so not directly comparable to our evaluations on the *ESEW* corpus, they are considerably higher, suggesting that our features are not doing as well as they could.

Indeed, following up on her thesis, Vajjala considered a set of 1000 sentence triples from the OSE corpus from among those used in our study. Using 10-fold cross validation, Vajjala (personal communication, 13 November 2015) has achieved 85.4% accuracy (ranging from 83.85% to 89.6% over the folds). This is not the exact same split of the data that we used, but is the same corpus, again suggesting that there is considerable room for improvement above what is offered by our features.

8 Discussion and Conclusion

8.1 Summary of Results

In this study we examined features for the ranking of sentences by their complexity, training linear models using psycholinguistically-motivated feature-sets on two corpora.

In line with earlier work on readability (see §2.1), our linear models are based on 'gold standard' annotations, with ranks based upon the published labels for the data (see §5.1). In the case of the English & Simple English Wikipedia (ESEW) corpus, the labels correspond to the notions of 'English' and 'Simple English', while for the One Stop English (OSE) corpus they correspond to the levels of difficulty believed appropriate by educators for 'Advanced', 'Intermediate', and 'Elementary' students of English. While the OSE corpus has been used by Vajjala (2015, inter alia), this is the first study to use the new ESEW corpus to evaluate models of readability.

We extracted four kinds of features rooted in psycholinguistic theories of sentence processing: surprisal, embedding depth, integration cost, and idea density. Surprisal (a measure of the unpredictability of an utterance in context) coupled with embedding depth (a measure of the amount of human memory required to parse a sentence) and our baseline features (average word length and sentence length) consistently performed as well as the full model. Integration cost (a measure of the difficulty of integrating new information with the sentence so far) and idea density (a measure of the number of propositions expressed per word) are generally less informative. This implies that there are not large gains to be had by using only the Stanford parser and these more easily extracted features. Instead, if these psycholinguistic measures are to be used, it is necessary to use the slower `ModelBlocks` parser to extract the more useful features.

A secondary consideration in this work was the benefit of sentence processing features for finer-grained distinctions between reading levels. Our analyses in this respect offer mixed results, sometimes indicating no special benefit for fine-grained distinctions and in other cases suggesting the opposite. Ultimately, our most important finding is simply that psycholinguistic measures of sentence processing do improve model accuracy by a couple of percentage points.

8.2 Directions for Future Work

The next step in this work is a collaboration, already in the planning stages, with Sowmya Vajjala. This collaboration will enable a comprehensive comparison of features and, more importantly, allow us to assess the necessity of extracting the features examined in this thesis from parser output. Vajjala's features (detailed in §2.3.1) rely on either dictionary lookups or single-best parser output, which are easier to acquire efficiently than the features reliant on `ModelBlocks`. Given our common interest in finding a good model of sentence complexity for use in automatic text simplification, we can also contribute more generally to theory in this area.

In this thesis we have limited our analysis of the features to use average and maximal values for each extracted feature. It is possible, however, that a better model would be to track the *change* in the value of each feature at each word as a subject reads the sentence. For example, the Uniform Information Density hypothesis (Jaeger, 2006) suggests that it may not be the average surprisal so much as the change in surprisal which is important.

Once we have a better model of sentential complexity, it would also be nice to validate this model in psycholinguistic experiments, completing the circuit. Our goal would be to examine both reading speed and comprehension accuracy in such an experiment.

8.3 Conclusion

Work in natural language generation, pedagogical tool development, and automatic text simplification can benefit from a scoring mechanism for individual sentences.

Based on psycholinguistic theories of sentence processing, we extracted sentence features shown to correlate with reading times. When incorporated into a simple linear ranking model, the improved performance by between 0.5 and 3 percentage points over a baseline model consistently scoring in the 70-80% range.

This is the first time these features have been examined in the context of sentence ranking, and our findings suggest that future work to incorporate these features into other NLP tasks will be worthwhile. Furthermore, these findings provide a kind of secondary evidence that these features are relevant to human perception of sentence difficulty, as they improve our ability to model the implicit categorical models of human writers when they aim to create a 'simpler' text.

In addition to these scientific contributions, the code for our implementation of the averaged perceptron model, as well as various scripts facilitating the use of `ModelBlocks` and the Stanford parser to extract these features, will be available from the author's website.

References

R. Harald Baayen, R. Piepenbrock, and L. Gulikers. 1995. The CELEX lexical databases.

Cati Brown, Tony Snodgrass, Susan J. Kemper, Ruth Herman, and Michael A. Covington. 2008. Automatic measurement of propositional idea density from part-of-speech tagging. *Behavior Research Methods*, 40(2):540–545.

CCSSO. 2010. Common core state standards for english language arts & literacy in history/social studies, science, and technical subjects. Appendix B: Text exemplars and sample performance tasks. Technical report, National Governors Association Center for Best Practices, Council of Chief State School Officers.

Jeanne S. Chall. 1958. *Readability: an appraisal of research and application.* The Ohio State University, Columbus, OH, USA.

Vineeta Chand, Kathleen Baynes, Lisa Bonnici, and Sarah Tomaszewski Farias. 2010. Analysis of Idea Density (AID): A Manual. Technical report, University of California at Davis.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and Methods for Text Simplification. In *Proc. of the 16th International Conference on Computational Linguistics, Proceedings of the Conference (COLING)*, Copenhagen, Denmark. Association for Computational Linguistics.

Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proc. of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar. Association for Computational Linguistics.

Meri Coleman and T. L. Liau. 1975. A Computer Readability Formula Designed for Machine Scoring. *Journal of Applied Psychology*, 60(2):283–284.

Michael Collins. 2002. Ranking Algorithms for Named–Entity Extraction: Boosting and the Voted Perceptron. In *Proc. of the 40th Annual Meeting of the Association*

for Computational Linguistics (ACL), pages 489–496, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks. *Machine Learning*, 20(3):273–297.

Scott A. Crossley, David B. Allen, and Danielle S. McNamara. 2011. Text readability and intuitive simplification: A comparison of readability formulas. *Reading in a Foreign Language*, 23(1):84–101.

Andre Luiz Verucci Da Cunha, Lucilene Bender De Sousa, Leticia Lessa Mansur, and Sandra Maria Aluísio. 2015. Automatic Proposition Extraction from Dependency Trees: Helping Early Prediction of Alzheimer's Disease from Narratives. *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, pages 127–130.

Edgar Dale and Jeanne S. Chall. 1948. A Formula for Predicting Readability. *Educational Research Bulletin*, 27(1):11–20+28.

Vera Demberg and Frank Keller. 2008. Data from Eye-tracking Corpora as Evidence for Theories of Syntactic Processing Complexity. *Cognition*, 109(2):193–210.

William H. Dubay. 2007. *Unlocking Language: The Classic Readability Studies.*

Lijun Feng, Noémie Elhadad, and Matt Huenerfauth. 2009. Cognitively motivated features for readability assessment. In *Proc. of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 229–237.

Rudolf Flesch. 1948. A new readability yardstick. *Journal of Applied Psychology*, 32(3):221–233.

Yoav Freund and Robert E. Schapire. 1999. Large Margin Classication Using the Perceptron Algorithm. *Machine Learning*, 37(3):277–296.

Edward Gibson. 2000. The Dependency Locality Theory: A Distance-Based Theory of Linguistic Complexity. In Y Miyashita, A Marantz, and W O'Neil, editors, *Image, Language, Brain*, chapter 5, pages 95–126. MIT Press, Cambridge, Massachusetts.

Daniel Gildea and David Temperley. 2010. Do Grammars Minimize Dependency Length? *Cognitive Science*, 34:286–310.

Arthur C. Graesser, Danielle S. McNamara, Max M. Louwerse, and Zhiqiang Cai. 2004. Coh-metrix: analysis of text on cohesion and language. *Behavior Research Methods, Instruments, & Computers*, 36(2):193–202.

William S. Gray and Bernice E. Leary. 1935. *What makes a book readable*. University of Chicago Press, Chicago, Illinois, USA.

John T. Hale. 2001. A Probabilistic Earley Parser as a Psycholinguistic Model. In *NAACL*.

Ralf Herbrich, Thore Graepel, and Klaus Obermayer. 1999. Support Vector Learning for Ordinal Regression A Risk Formulation for Ordinal Regression. *Proc. of the 9th International Conference on Artificial Neural Networks*, pages 97–102.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning Sentences from Standard Wikipedia to Simple Wikipedia. In *Proc. of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, Denver, Colorado, USA.

T. Florian Jaeger. 2006. *Redundancy and Syntactic Reduction in Spontaneous Speech*. Unpublished dissertation, Stanford University.

T. Florian Jaeger. 2010. Redundancy and reduction: speakers manage syntactic information density. *Cognitive Psychology*, 61(1):23–62, aug.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. *Proc. of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142.

Thorsten Joachims. 2006. Training linear SVMs in linear time. *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, page 217.

David Kauchak, Obay Mouradi, Christopher Pentoney, and Gondy Leroy. 2014. Text simplification tools: Using machine learning to discover features that identify difficult text. *Proceedings of the Annual Hawaii International Conference on System Sciences*, pages 2616–2625.

Alan Kennedy and Joël Pynte. 2005. Parafoveal-on-foveal effects in normal reading. *Vision Research*, 45(2):153–168.

J. Peter Kincaid, Robert P. Fishburne, Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (Automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy enlisted personnel. Technical report, Naval Technical Training Command, Memphis - Millington, TN, USA.

James R. King. 2012. A critical review of proposition analysis in Alzheimer's research and elsewhere. *Linguistics and Education*, 23(4):388–401.

Walter Kintsch and Janice Keenan. 1973. Reading Rate and of Propositions Retention as a Function of the Number in the Base Structure of Sentences. *Cognitive Psychology*, 5:257–274.

Walter Kintsch. 1972. Notes on the structure of semantic memory. In Endel Tulving and Wayne Donaldson, editors, *Organization of memory*, pages 247–308. Academic Press, New York, New York, USA.

Roger Levy and T. Florian Jaeger. 2007. Speakers optimize information density through syntactic reduction. *Advances in Neural Information Processing Systems 20 (NIPS)*.

Roger Levy. 2008. Expectation-based syntactic comprehension. *Cognition*, 106(3):1126–77, mar.

Bertha A. Lively and S. L. Pressey. 1923. A Method for Measuring the 'Vocabulary Burden' of Textbooks. *Educational Administration and Supervision*, IX:389–398.

Macmillan Publishers. 2015. onestopenglish.

Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157.

George A. Miller. 1956a. Human memory and the storage of information. *IRE Transactions on Information Theory (IT-2)*, 2(3):129–137, sep.

George A. Miller. 1956b. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological Review*.

Courtney Napoles and Mark Dredze. 2010. Learning Simple Wikipedia : A Cogitation in Ascertaining Abecedarian Language. *Computational Linguistics*, (June):42–50.

Charles A. Perfetti. 1969. Lexical density and phrase structure depth as variables in sentence retention. *Journal of Verbal Learning and Verbal Behavior*, 8(6):719–724.

Sarah E. Petersen and Mari Ostendorf. 2009. A machine learning approach to reading level assessment. *Computer Speech and Language*, 23:89–106.

Sarah E. Petersen. 2007. *Natural Language Processing Tools for Reading Level Assessment and Text Simplification for Bilingual Education*. {PhD} thesis, University of Washington.

Brian Roark, Asaf Bachrach, Carlos Cardenas, and Christophe Pallier. 2009. Deriv-

ing lexical and syntactic expectation-based measures for psycholinguistic modeling via incremental top-down parsing. *EMNLP*.

Claude E. Shannon. 1948. A Mathematical Theory of Communication. *The Bell System Technical Journal*, 27(3):379–423.

L. A. Sherman. 1893. *Analytics of Literature*. Ginn & Company, Boston, Massachusetts, USA.

Advaith Siddharthan. 2004. *Syntactic simplification and text cohesion*. Technical report; dissertation, University of Cambridge, Gonville and Caius College, dec.

E. A. Smith and R. J. Senter. 1967. Automated readability index. Technical report, Aerospace Medical Research Laboratories, Aerospace Medical Division, Air Force Systems Command, Wright-Patterson Air Force Base, Ohio, USA.

Althea Turner and Edith Greene. 1977. The Construction and Use of a Propositional Text Base. Technical report, Institute for the Study of Intellectual Behavior, University of Colorado, Boulder, CO.

Sowmya Vajjala and Detmar Meurers. 2012. On Improving the Accuracy of Readability Classification using Insights from Second Language Acquisition. In *Proceedings of the 7th Workshop on Innovative Use of NLP for Building Educational Applications (BEA7)*. Association for Computational Linguistics.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. In *Proc. of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Gothenburg, Sweden. Association for Computational Linguistics.

Sowmya Vajjala. 2015. *Analyzing Text Complexity and Text Simplification: Connecting Linguistics, Processing and Educational Applications*. {PhD} thesis, Eberhard Karls Universitaet Tuebingen.

Marten van Schijndel and William Schuler. 2012. Connectionist-Inspired Incremental PCFG Parsing. In *Proc. of the 3rd Workshop on Cognitive Modeling and Computational Linguistics (CMCL)*.

Marten van Schijndel, Andy Exley, and William Schuler. 2013. A model of language processing as hierarchic sequential prediction. *Topics in Cognitive Science*, 5(3):522–40.

Wikipedia. 2015a. Basic English.

Wikipedia. 2015b. Simple Wikipedia Main Page.

Michael D. Wilson. 1988. The MRC Psycholinguistic Database: Machine Readable Dictionary, Version 2. *Behavior Research Methods, Instruments, & Computers*, 20(1):6–11.

Stephen Wu, Asaf Bachrach, Carlos Cardenas, and William Schuler. 2010. Complexity metrics in an incremental right-corner parser. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1189–1198.

Victor H. Yngve. 1960. A Model and an Hypothesis for Language Structure. *American Philosophical Society*, 104(5):444–466.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A Monolingual Tree-based Translation Model for Sentence Simplification. In Chu-Ren Huang and Dan Jurafsky, editors, *Proc. of the 23rd International Conference on Computational Linguistics (COLING)*, pages 1353–1361, Beijing, China. Association for Computational Linguistics.

A Software for Feature Extraction

The psycholinguistic measures we extract are based on sentence structure and theories of its processing. To this end we use two different parsers: the ModelBlock's parser (van Schijndel et al., 2013) and the Stanford Dependency Parser (Chen and Manning, 2014). This appendix details the basic software requirements, while our scripts will be made available at `http://www.davehowcroft.com`, along with further documentation.

A.1 Software Requirements

You will need to install the `ModelBlocks` parser, the Stanford Parser, and the `extended_ptb_tokenizer`. Our shell scripts were written for `bash` and Python 3.4.[1] In addition to `bash`, my scripts also make use of `awk` and `sed`.

`ModelBlocks` is available on SourceForge at `http://sourceforge.net/projects/modelblocks/`. The project relies on GNU Make to build the parser and execute various commands for parsing and data extraction.

The **Stanford Parser** is available from `http://www-nlp.stanford.edu/software/` either as a standalone project or as part of the Stanford CoreNLP toolkit. We used a fresh installation of version 3.5.2 (release date: 20 April 2015) for the extraction of propositional idea density and version 1.3.2 (release date: 22 May 2015) for integration cost. The codebase for `icy-parses` is the reason for this: the original version of the code integrated a special Java wrapper for the Stanford Parser which adds to the complexity of updating the code to work with a new version of the Stanford Parser.

[1] Although we have tried to keep our Python scripts backwards-compatible with Python 2.7, we make no guarantee that they will work in this environment.

The `extended_ptb_tokenizer` is available on GitHub at `https://www.github.com/vansky/extended_penn_tokenizer`. It is a `sed` script.

B Code for the Analysis

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

"""AveragedPerceptron model built using numpy"""

import logging
import pickle
import numpy as np
from scipy import stats
from math import sqrt, factorial
from itertools import combinations

# Define formats for logging
log_msg_format = "%(message)s"
# Configure logging; use level=logging.DEBUG for debugging
logging.basicConfig(format=log_msg_format, level=logging.INFO
    ↪ )
logging.debug("Finished primary imports.")

try:
    from termcolor import colored
except ImportError:
    def colored(text, color=None, on_color=None, attributes=
        ↪ None):
        return text
    logging.debug("ImportError on 'from termcolor import
        ↪ colored'.")

__author__ = "David M. Howcroft (dave.howcroft@gmail.com)"

data_directory = "../data/"
```

```
class AveragedPerceptron(object):
    def __init__(self, samples=None, view=None):
        """Averaged perceptron learner"""
        logging.debug("Initializing AveragedPerceptron with
           ↪ samples={} and view={}".format(samples, view))

        if samples:
            self._samples = samples
        else:
            self._samples = None
        self._view = view
        self._weights = None
        self._avg_weights = None
        self._cv_weights = None
        self._cv_avg_weights = None
        if self.data is not None:
            self.zero_weights(len(self.data[0]))
            if self.labels is not None:
                self.fit()

    @property
    def data(self):
        return self._samples.data_by_view(self._view)

    @property
    def labels(self):
        return self._samples.labels

    @property
    def num_rows(self):
        return self._samples.num_rows

    @property
    def num_columns(self):
        return self._samples.num_columns

    @property
    def shape(self):
        return self._samples.shape

    def zero_weights(self, number_of_dimensions):
```

```
        self._avg_weights = np.zeros(number_of_dimensions)
        self._weights = np.zeros(number_of_dimensions)

    @staticmethod
    def _train_weights(data_labels, weights, avg_weights, num_iterations=100):
        for iteration in range(1, num_iterations + 1):
            logging.debug("Beginning iteration {} of {}".format(iteration, num_iterations))
            for datum, label in data_labels:
                predicted_label = np.sign(weights.dot(datum))
                if predicted_label != label:
                    weights += (label - predicted_label) * datum
                avg_weights += weights
        return weights, avg_weights

    def fit(self):
        if self._weights is None or self._avg_weights is None:
            self.zero_weights(self.num_columns)
        data_labels = list(zip(self.data, self.labels))
        weights, avg_weights = self._weights, self._avg_weights
        self._weights, self._avg_weights = AveragedPerceptron._train_weights(data_labels, weights, avg_weights)

    def cross_validate_by_qid(self, num_folds=10):
        if not self._samples.group_size:
            return self.cross_validate(num_folds)
        if self._cv_weights is None or self._cv_avg_weights is None:
            num_dimensions = len(self.data[0])
            self._cv_weights = [np.zeros(num_dimensions) for _ in range(0, num_folds)]
            self._cv_avg_weights = [np.zeros(num_dimensions) for _ in range(0, num_folds)]

        scores = []
        data_labels = list(zip(self.data, self.labels))
```

```
        group_size = self._samples.group_size
        logging.debug("Group size {}".format(group_size))

        predicted_labels = np.zeros(len(self.labels))
        for fold in range(1, num_folds+1):
            logging.debug("Beginning fold {} of {}".format(
               ↪ fold, num_folds))
            # Training
            weights, avg_weights = self._cv_weights[fold-1],
               ↪ self._cv_avg_weights[fold-1]
            training_data = []
            for item_count, item in enumerate(data_labels):
                if (item_count % group_size) % num_folds !=
                   ↪ fold - 1:
                    training_data.append(item)
            weights, avg_weights = AveragedPerceptron.
               ↪ _train_weights(training_data, weights,
               ↪ avg_weights)

            # Testing
            item_count, correct, tested = 0, 0, 0
            for datum, label in data_labels:
                if (item_count % group_size) % num_folds ==
                   ↪ fold - 1:
                    tested += 1
                    predicted_label = np.sign(avg_weights.dot
                       ↪ (datum))
                    predicted_labels[item_count] =
                       ↪ predicted_label
                    if label == predicted_label:
                        correct += 1
                item_count += 1
            logging.debug("Tested {} items".format(tested))
            scores.append(correct / tested)

            self._cv_weights[fold-1], self._cv_avg_weights[
               ↪ fold-1] = weights, avg_weights
        return sum(scores) / num_folds, predicted_labels

    def score(self, data=None, labels=None):
        if not data:
```

```
            data = self.data
        if not labels:
            labels = self.labels

        correct = 0
        predictions = np.zeros(len(labels))
        for index, datum, label in enumerate(zip(data, labels
            ↪ )):
            predicted_label = self._avg_weights.dot(datum)
            if label == np.sign(predicted_label):
                correct += 1
            predictions[index] = predicted_label

        return correct / len(data), predictions
```

C Settings for using SVM^{rank}

For each combination of corpus and model we used the following c values in SVM^{rank}.

```
esew/baseline 20.0
esew/surprisal 50.0
esew/embedding 100.0
esew/integration 50.0
esew/idea_density 0.1
esew/psycholing 100.0
esew/modelblocks 100.0
esew/stanford 100.0
esew/baseline_modelblocks 20.0
esew/baseline_stanford 100.0
esew/full_model 100.0
ose/baseline 0.01
ose/surprisal 0.5
ose/embedding 0.01
ose/integration 10.0
ose/idea_density 0.01
ose/psycholing 0.5
ose/modelblocks 10.0
ose/stanford 10.0
ose/baseline_modelblocks 0.01
ose/baseline_stanford 0.01
ose/full_model 0.5
```

D Significance Tables

This chapter contains all of the significance tables for the different model comparisons. Note that all of our significance testing is within-datasets and therefore within columns rather than across columns.

D.1 Notation

The data presented in this appendix follows a regular format. Each analysis starts out by reporting the accuracies for that dataset, accompanied by a caption describing the analysis as well as comments on which differences are significant. Each such table is then followed by a series of smaller tables reporting (1) p-values, (2) significance in standard "star" notation, and (3) the effect sizes. A description of which analyses were done and why is found in §7.2.

The p-values are reported for completeness. The significance tables contain one star (*) for significance at the $p < 0.05$ level and two stars (**) for the $p < 0.01$ level, these levels being calculated based on the p-values modified by Bonferroni correction. Finally, the effect sizes are reported so that we can have an idea of how substantial the significant differences we report are.

In the accuracy tables starting each section, the best value in each column is bolded and we have used the following notation convention:

- When no item in a column has any marking, all differences are significant.
- When a pair of models are significantly different from each other, but no other differences are significant, we mark the values with a preceding +.

- When a pair of models are *not* significantly different from each other, but all other differences *are* significant, we mark the pair with preceding − signs.

- When the four models form groups of two, where each pair is significantly different from the other but the items in the pair are not significantly different from each other, we mark each pair with † or ‡, with items in the same pair sharing the same symbol.

- When a group of models are not significantly different from one another, we have marked the 'statistically equivalent' values with a preceding =.

- Finally, when adjacent pairs of items differ significantly from one another, we have used ⟨ to highlight the significance of that difference.

D.2 Significance Testing for Averaged Perceptron Models

D.2.1 Significance with corrections for hypothesis testing

D.2.1.1 Non-normalized difference features

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Baseline	71.24	‡75.10	+82.11	‡71.36
Baseline+ModelBlocks	72.58	**†77.94**	**+85.29**	**†74.61**
Baseline+Stanford	72.39	‡75.50	82.87	‡71.85
FullModel	**73.22**	†77.24	84.38	†74.23

Table D.1: Duplicate of Table 7.1: Averaged Perceptron performance on difference features using 10-fold cross-validation. All differences are significant in the $ESEW$ column. In OSE_{All} and OSE_{Close} the difference between items marked with the same symbol are not different, but all others are. For $OSE_{Adv-Elem}$ only the difference between the Baseline and the Baseline+ModelBlocks systems is significant. (This is denoted with the + sign.)

	Base	Base+MB	Base+Stan	Full
Baseline	-	4.6e-07	3.5e-19	2e-26
Baseline+ModelBlocks	4.6e-07	-	4.4e-05	1.6e-14
Baseline+Stanford	3.5e-19	4.4e-05	-	0.00058
FullModel	2e-26	1.6e-14	0.00058	-

Table D.2: *ESEW* p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	**	**
Baseline+ModelBlocks	**	-	**	**
Baseline+Stanford	**	**	-	**
FullModel	**	**	**	-

Table D.3: *ESEW* significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.02	0.03	0.03
Baseline+ModelBlocks	0.02	-	0.01	0.02
Baseline+Stanford	0.03	0.01	-	0.01
FullModel	0.03	0.02	0.01	-

Table D.4: *ESEW* effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	1e-06	0.73	0.0036
Baseline+ModelBlocks	1e-06	-	1.6e-05	0.045
Baseline+Stanford	0.73	1.6e-05	-	0.0011
FullModel	0.0036	0.045	0.0011	-

Table D.5: OSE_{All} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	-	*
Baseline+ModelBlocks	**	-	**	-
Baseline+Stanford	-	**	-	**
FullModel	*	-	**	-

Table D.6: OSE_{All} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.08	-	0.05
Baseline+ModelBlocks	0.08	-	0.07	-
Baseline+Stanford	-	0.07	-	0.05
FullModel	0.05	-	0.05	-

Table D.7: OSE_{All} effect-size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.00011	0.29	0.01
Baseline+ModelBlocks	0.00011	-	0.0018	0.1
Baseline+Stanford	0.29	0.0018	-	0.031
FullModel	0.01	0.1	0.031	-

Table D.8: $OSE_{Adv-Elem}$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	-	-
Baseline+ModelBlocks	**	-	*	-
Baseline+Stanford	-	*	-	-
FullModel	-	-	-	-

Table D.9: $OSE_{Adv-Elem}$ significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.11	-	-
Baseline+ModelBlocks	0.11	-	0.09	-
Baseline+Stanford	-	0.09	-	-
FullModel	-	-	-	-

Table D.10: $OSE_{Adv-Elem}$ effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	1.5e-05	0.8	0.002
Baseline+ModelBlocks	1.5e-05	-	0.00023	0.4
Baseline+Stanford	0.8	0.00023	-	0.00046
FullModel	0.002	0.4	0.00046	-

Table D.11: OSE_{Close} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	-	*
Baseline+ModelBlocks	**	-	**	-
Baseline+Stanford	-	**	-	**
FullModel	*	-	**	-

Table D.12: OSE_{Close} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.08	-	0.06
Baseline+ModelBlocks	0.08	-	0.07	-
Baseline+Stanford	-	0.07	-	0.07
FullModel	0.06	-	0.07	-

Table D.13: OSE_{Close} effect size

D.2.1.2 z-score Normalized Difference Features

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Baseline	73.37	=78.14	=84.69	74.64
Baseline+ModelBlocks	72.91	**=78.87**	**=85.29**	75.81
Baseline+Stanford	**−73.81**	76.93	=83.63	+73.97
FullModel	−73.76	=78.32	=84.53	**+76.04**

Table D.14: Duplicate of Table 7.2: Averaged Perceptron results for z-score normalized difference features. For $ESEW$, all differences except the one marked by − are significant. For OSE_{All}, only Baseline+Stanford is significantly different from the other models. For $OSE_{Adv-Elem}$, no differences are significant. For OSE_{Close}, only the difference between Baseline+Stanford and FullModel are significant. Values within a column marked with = are not significantly different from each other.

	Base	Base+MB	Base+Stan	Full
Baseline	-	1.5e-06	3.1e-06	0.00019
Baseline+ModelBlocks	1.5e-06	-	1.8e-15	1.2e-21
Baseline+Stanford	3.1e-06	1.8e-15	-	0.47
FullModel	0.00019	1.2e-21	0.47	-

Table D.15: $ESEW$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	**	**
Baseline+ModelBlocks	**	-	**	**
Baseline+Stanford	**	**	-	-
FullModel	**	**	-	-

Table D.16: $ESEW$ significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.02	0.01	0.01
Baseline+ModelBlocks	0.02	-	0.03	0.03
Baseline+Stanford	0.01	0.03	-	-
FullModel	0.01	0.03	-	-

Table D.17: $ESEW$ effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.15	0.0022	0.73
Baseline+ModelBlocks	0.15	-	0.0005	0.11
Baseline+Stanford	0.0022	0.0005	-	0.0063
FullModel	0.73	0.11	0.0063	-

Table D.18: OSE_{All} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	*	-
Baseline+ModelBlocks	-	-	**	-
Baseline+Stanford	*	**	-	*
FullModel	-	-	*	-

Table D.19: OSE_{All} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	0.05	-
Baseline+ModelBlocks	-	-	0.06	-
Baseline+Stanford	0.05	0.06	-	0.04
FullModel	-	-	0.04	-

Table D.20: OSE_{All} effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.45	0.09	0.86
Baseline+ModelBlocks	0.45	-	0.054	0.21
Baseline+Stanford	0.09	0.054	-	0.27
FullModel	0.86	0.21	0.27	-

Table D.21: $OSE_{Adv-Elem}$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	-	-
Baseline+ModelBlocks	-	-	-	-
Baseline+Stanford	-	-	-	-
FullModel	-	-	-	-

Table D.22: $OSE_{Adv-Elem}$ significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	-	-
Baseline+ModelBlocks	-	-	-	-
Baseline+Stanford	-	-	-	-
FullModel	-	-	-	-

Table D.23: $OSE_{Adv-Elem}$ effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.062	0.2	0.034
Baseline+ModelBlocks	0.062	-	0.0071	0.62
Baseline+Stanford	0.2	0.0071	-	0.00073
FullModel	0.034	0.62	0.00073	-

Table D.24: OSE_{Close} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	-	-
Baseline+ModelBlocks	-	-	*	-
Baseline+Stanford	-	*	-	**
FullModel	-	-	**	-

Table D.25: OSE_{Close} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	-	-
Baseline+ModelBlocks	-	-	0.05	-
Baseline+Stanford	-	0.05	-	0.07
FullModel	-	-	0.07	-

Table D.26: OSE_{Close} effect size

D.2.1.3 Non-normalized Difference Features with Interactions

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
BASELINE	72.20	−76.71	=84.98	72.68
BASELINE+MODELBLOCKS	**72.71**	**79.62**	**=86.87**	**76.08**
BASELINE+STANFORD	70.99	75.40	82.65	71.70
FULLMODEL	72.20	−78.57	=86.42	75.10

Table D.27: Duplicate of Table 7.3: Averaged Perceptron performance on difference features with added interaction terms using 10-fold cross-validation. All models are significantly different on the $ESEW$ dataset, including the two models with identical performance. The only difference that is not significant for OSE_{All} is that between the BASELINE and the FULLMODEL. For $OSE_{Adv-Elem}$, only the BASELINE+STANFORD model is significantly different from the other models. For the OSE_{Close} dataset, each model is significantly different only from the next model in the table. That is, BASELINE differs from BASELINE+MODELBLOCKS which differs from BASELINE+STANFORD which differs from FULLMODEL, but no other differences are significant.

	BASE	BASE+MB	BASE+STAN	FULL
BASELINE	-	8.3e-07	1.6e-41	2.6e-16
BASELINE+MODELBLOCKS	8.3e-07	-	3.1e-18	7.2e-06
BASELINE+STANFORD	1.6e-41	3.1e-18	-	5e-09
FULLMODEL	2.6e-16	7.2e-06	5e-09	-

Table D.28: $ESEW$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	**	**
Baseline+ModelBlocks	**	-	**	**
Baseline+Stanford	**	**	-	**
FullModel	**	**	**	-

Table D.29: *ESEW* significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.02	0.04	0.03
Baseline+ModelBlocks	0.02	-	0.03	0.01
Baseline+Stanford	0.04	0.03	-	0.02
FullModel	0.03	0.01	0.02	-

Table D.30: *ESEW* effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	5e-06	0.00092	0.03
Baseline+ModelBlocks	5e-06	-	9.5e-12	0.0033
Baseline+Stanford	0.00092	9.5e-12	-	1.2e-08
FullModel	0.03	0.0033	1.2e-08	-

Table D.31: OSE_{All} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	**	-
Baseline+ModelBlocks	**	-	**	*
Baseline+Stanford	**	**	-	**
FullModel	-	*	**	-

Table D.32: OSE_{All} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.07	0.05	-
Baseline+ModelBlocks	0.07	-	0.11	0.05
Baseline+Stanford	0.05	0.11	-	0.09
FullModel	-	0.05	0.09	-

Table D.33: OSE_{All} effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.036	0.00027	0.14
Baseline+ModelBlocks	0.036	-	3.9e-06	0.42
Baseline+Stanford	0.00027	3.9e-06	-	2.4e-06
FullModel	0.14	0.42	2.4e-06	-

Table D.34: $OSE_{Adv-Elem}$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	**	-
Baseline+ModelBlocks	-	-	**	-
Baseline+Stanford	**	**	-	**
FullModel	-	-	**	-

Table D.35: $OSE_{Adv-Elem}$ significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	0.10	-
Baseline+ModelBlocks	-	-	0.13	-
Baseline+Stanford	0.10	0.13	-	0.13
FullModel	-	-	0.13	-

Table D.36: $OSE_{Adv-Elem}$ effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	5.9e-05	0.037	0.026
Baseline+ModelBlocks	5.9e-05	-	1.1e-07	0.051
Baseline+Stanford	0.037	1.1e-07	-	5.2e-06
FullModel	0.026	0.051	5.2e-06	-

Table D.37: OSE_{Close} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	**	-	-
Baseline+ModelBlocks	**	-	**	-
Baseline+Stanford	-	**	-	**
FullModel	-	-	**	-

Table D.38: OSE_{Close} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.08	-	-
Baseline+ModelBlocks	0.08	-	0.10	-
Baseline+Stanford	-	0.10	-	0.09
FullModel	-	-	0.09	-

Table D.39: OSE_{Close} effect size

D.2.1.4 z-score normalized difference features with interactions

Model	$ESEW$	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Baseline	−73.79	78.74	†85.51	75.06
Baseline+ModelBlocks	−73.71	**79.90**	‡**87.93**	+**76.23**
Baseline+Stanford	74.47	77.04	†84.98	+73.89
FullModel	**75.09**	78.19	‡**87.93**	75.13

Table D.40: Duplicate of 7.4: Averaged Perceptron performance on z-score normalized difference features with added interaction terms using 10-fold cross-validation. For $ESEW$, the only non-significant difference is that between the Baseline and FullModels. For OSE_{All}, Baseline+ModelBlocks is significantly different from Baseline+Stanford and FullModel, but not from the Baseline. Meanwhile, Baseline+Stanford is also significantly different from the Baseline. For $OSE_{Adv-Elem}$, we again get a pattern where Baseline and Baseline+Stanford are statistically indistinguishable but are significantly different from the other pair of models, which are in turn statistically 'equivalent' to each other. Finally, the only significant difference for OSE_{Close} is that between the Baseline+ModelBlocks and the Baseline+Stanford models.

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.34	2.5e-10	5.8e-35
Baseline+ModelBlocks	0.34	-	2.9e-12	3.6e-47
Baseline+Stanford	2.5e-10	2.9e-12	-	7.6e-16
FullModel	5.8e-35	3.6e-47	7.6e-16	-

Table D.41: $ESEW$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	**	**
Baseline+ModelBlocks	-	-	**	**
Baseline+Stanford	**	**	-	**
FullModel	**	**	**	-

Table D.42: *ESEW* significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	0.02	0.04
Baseline+ModelBlocks	-	-	0.02	0.05
Baseline+Stanford	0.02	0.02	-	0.03
FullModel	0.04	0.05	0.03	-

Table D.43: *ESEW* effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.041	0.00026	0.34
Baseline+ModelBlocks	0.041	-	4.4e-06	0.00027
Baseline+Stanford	0.00026	4.4e-06	-	0.036
FullModel	0.34	0.00027	0.036	-

Table D.44: OSE_{All} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	**	-
Baseline+ModelBlocks	-	-	**	**
Baseline+Stanford	**	**	-	-
FullModel	-	**	-	-

Table D.45: OSE_{All} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	0.06	-
Baseline+ModelBlocks	-	-	0.07	0.06
Baseline+Stanford	0.06	0.07	-	-
FullModel	-	0.06	-	-

Table D.46: OSE_{All} effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.0057	0.43	0.0041
Baseline+ModelBlocks	0.0057	-	0.0016	-
Baseline+Stanford	0.43	0.0016	-	0.00019
FullModel	0.0041	-	0.00019	-

Table D.47: $OSE_{Adv-Elem}$ p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	*	-	*
Baseline+ModelBlocks	*	-	**	-
Baseline+Stanford	-	**	-	**
FullModel	*	-	**	-

Table D.48: $OSE_{Adv-Elem}$ significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.08	-	0.08
Baseline+ModelBlocks	0.08	-	0.09	-
Baseline+Stanford	-	0.09	-	0.10
FullModel	0.08	-	0.10	-

Table D.49: $OSE_{Adv-Elem}$ effect size

	Base	Base+MB	Base+Stan	Full
Baseline	-	0.11	0.059	0.92
Baseline+ModelBlocks	0.11	-	0.0031	0.075
Baseline+Stanford	0.059	0.0031	-	0.082
FullModel	0.92	0.075	0.082	-

Table D.50: OSE_{Close} p-values

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	-	-
Baseline+ModelBlocks	-	-	*	-
Baseline+Stanford	-	*	-	-
FullModel	-	-	-	-

Table D.51: OSE_{Close} significance

	Base	Base+MB	Base+Stan	Full
Baseline	-	-	-	-
Baseline+ModelBlocks	-	-	0.06	-
Baseline+Stanford	-	0.06	-	-
FullModel	-	-	-	-

Table D.52: OSE_{Close} effect size

D.2.2 Significance with corrections for data exploration

In this section we abbreviate each of the feature sets to save space. Items are listed in the same order as in the tables in §7.3.2 and the abbreviations should be transparent.

D.2.2.1 Non-normalized difference features

Model	ESEW	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Surprisal	61.48	62.40	$=^{1}$67.25	$=^{1,2}$59.74
Embedding	$=^{1}$65.04	64.93	$=^{1,2}$71.70	$=^{1}$61.82
IntegrationCost	$=^{2}$63.84	$=^{1}$67.27	$=^{3}$76.46	$=^{3}$62.27
IdeaDensity	55.33	50.97	54.50	$=^{2}$48.87
Psycholinguistic	69.76	$=^{2}$75.07	$=^{3,4}$80.99	$=^{4}$71.89
ModelBlocks	$=^{1}$65.57	$=^{1}$71.05	$=^{2,3}$74.42	$=^{3}$67.36
Stanford	$=^{2}$64.68	$=^{1}$67.98	$=^{3}$77.51	$=^{3}$62.98
Baseline	71.24	$=^{2}$75.10	$=^{4}$82.11	$=^{4}$71.36
Baseline+ModelBlocks	72.58	**77.94**	**85.29**	**74.61**
Baseline+Stanford	72.39	$=^{2}$75.50	82.87	$=^{4}$71.85
FullModel	**73.22**	77.24	84.38	74.23

Table D.53: Duplicate of Table 7.5: Averaged Perceptron performance on difference features using 10-fold cross-validation. Differences among the four models used for the main hypothesis are not explored here. Their accuracies are reported in order to compare with the feature subsets. All differences except those marked with = are significant. Those marked with = are further annotated with a superscript to indicate equivalence classes. Such classes are valid only within a single column. For example, the difference between Embedding and ModelBlocks is not significant for $ESEW$, though it is for OSE_{All} and OSE_{Close}. For OSE_{All}, ModelBlocks, IntegrationCost, and Stanford are not significantly different from each other. The overall Psycholinguistic model is also not different from the Baseline or Baseline+Stanford models.

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	9.4e-78	3.6e-233	6.6e-15	0.0	7.4e-299	4.5e-251	0.0	0.0	0.0	0.0
Emb	9.4e-78	-	4.5e-59	5.5e-157	5.2e-182	0.0014	6.4e-70	0.0	0.0	0.0	0.0
IC	3.6e-233	4.5e-59	-	0.0	1.3e-46	1.5e-40	0.0036	2e-146	5.3e-165	1.2e-269	2e-272
ID	6.6e-15	5.5e-157	0.0	-	0.0	7e-165	0.0	0.0	0.0	0.0	0.0
Psych	0.0	5.2e-182	1.3e-46	0.0	-	3.3e-183	2.8e-37	1.9e-61	2.4e-96	2.9e-146	5e-200
MB	7.4e-299	0.0014	1.5e-40	7e-165	3.3e-183	-	2.9e-48	0.0	0.0	0.0	0.0
Stan	4.5e-251	6.4e-70	0.0036	0.0	2.8e-37	2.9e-48	-	5.2e-134	7.2e-152	5.2e-258	2.4e-259
Base	0.0	0.0	2e-146	0.0	1.9e-61	0.0	5.2e-134	-	4.6e-07	3.5e-19	2e-26
B+MB	0.0	0.0	5.3e-165	0.0	2.4e-96	0.0	7.2e-152	4.6e-07	-	4.4e-05	1.6e-14
B+Stan	0.0	0.0	1.2e-269	0.0	2.9e-146	0.0	5.2e-258	3.5e-19	4.4e-05	-	0.00058
Full	0.0	0.0	2e-272	0.0	5e-200	0.0	2.4e-259	2e-26	1.6e-14	0.00058	-

Table D.54: *ESEW* p-values

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	**	**	**	**	**	**	**	**	**	**
Emb	**	-	**	**	**	-	**	**	**	**	**
IC	**	**	-	**	**	**	-	**	**	**	**
ID	**	**	**	-	**	**	**	**	**	**	**
Psych	**	**	**	**	-	**	**	**	**	**	**
MB	**	-	**	**	**	-	**	**	**	**	**
Stan	**	**	-	**	**	**	-	**	**	**	**
Base	**	**	**	**	**	**	**	-	**	**	**
B+MB	**	**	**	**	**	**	**	**	-	**	**
B+Stan	**	**	**	**	**	**	**	**	**	-	*
Full	**	**	**	**	**	**	**	**	**	*	-

Table D.55: *ESEW* significance

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	0.06	0.10	0.02	0.16	0.12	0.11	0.18	0.20	0.20	0.21
Emb	0.06	-	0.05	0.09	0.09	-	0.06	0.15	0.15	0.16	0.16
IC	0.10	0.05	-	0.14	0.05	0.04	-	0.08	0.09	0.11	0.11
ID	0.02	0.09	0.14	-	0.16	0.09	0.14	0.20	0.20	0.21	0.22
Psych	0.16	0.09	0.05	0.16	-	0.09	0.04	0.05	0.07	0.08	0.10
MB	0.12	-	0.04	0.09	0.09	-	0.05	0.13	0.15	0.14	0.16
Stan	0.11	0.06	-	0.14	0.04	0.05	-	0.08	0.08	0.11	0.11
Base	0.18	0.15	0.08	0.20	0.05	0.13	0.08	-	0.02	0.03	0.03
B+MB	0.20	0.15	0.09	0.20	0.07	0.15	0.08	0.02	-	0.01	0.02
B+Stan	0.20	0.16	0.11	0.21	0.08	0.14	0.11	0.03	0.01	-	0.01
Full	0.21	0.16	0.11	0.22	0.10	0.16	0.11	0.03	0.02	0.01	-

Table D.56: *ESEW* effect size

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	0.014	2.9e-19	1.3e-07	2.4e-43	4.9e-34	7.3e-19	7.6e-39	9.1e-59	3.1e-39	1.1e-53
Emb	0.014	-	2.7e-14	3.6e-19	2.9e-33	1e-13	1.7e-14	9.2e-40	3e-53	8.1e-40	1.5e-47
IC	2.9e-19	2.7e-14	-	9.7e-54	1.1e-06	0.42	0.93	4.8e-09	2.6e-16	5.1e-11	2.1e-15
ID	1.3e-07	3.6e-19	9.7e-54	-	3.2e-72	2.1e-43	9.6e-54	7.2e-86	1.9e-98	7.3e-85	3.5e-93
Psych	2.4e-43	2.9e-33	1.1e-06	3.2e-72	-	4.8e-08	4.3e-07	0.32	6.4e-06	0.21	9e-05
MB	4.9e-34	1e-13	0.42	2.1e-43	4.8e-08	-	0.46	4.4e-08	2e-18	1.9e-08	3.1e-15
Stan	7.3e-19	1.7e-14	0.93	9.6e-54	4.3e-07	0.46	-	2.1e-09	4.7e-17	1.1e-11	3.7e-16
Base	7.6e-39	9.2e-40	4.8e-09	7.2e-86	0.32	4.4e-08	2.1e-09	-	1e-06	0.73	0.0036
B+MB	9.1e-59	3e-53	2.6e-16	1.9e-98	6.4e-06	2e-18	4.7e-17	1e-06	-	1.6e-05	0.045
B+Stan	3.1e-39	8.1e-40	5.1e-11	7.3e-85	0.21	1.9e-08	1.1e-11	0.73	1.6e-05	-	0.0011
Full	1.1e-53	1.5e-47	2.1e-15	3.5e-93	9e-05	3.1e-15	3.7e-16	0.0036	0.045	0.0011	-

Table D.57: OSE_{All} p-values

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	-	**	**	**	**	**	**	**	**	**
Emb	-	-	**	**	**	**	**	**	**	**	**
IC	**	**	-	**	**	-	-	**	**	**	**
ID	**	**	**	-	**	**	**	**	**	**	**
Psych	**	**	**	**	-	**	**	-	**	-	**
MB	**	**	-	**	**	-	-	**	**	**	**
Stan	**	**	-	**	**	-	-	**	**	**	**
Base	**	**	**	**	-	**	**	-	**	-	-
B+MB	**	**	**	**	**	**	**	**	-	**	-
B+Stan	**	**	**	**	-	**	**	-	**	-	-
Full	**	**	**	**	**	**	**	-	-	-	-

Table D.58: OSE_{All} significance

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	-	0.14	0.08	0.22	0.19	0.14	0.21	0.26	0.21	0.24
Emb	-	-	0.12	0.14	0.19	0.12	0.12	0.21	0.24	0.21	0.23
IC	0.14	0.12	-	0.24	0.08	-	-	0.09	0.13	0.10	0.13
ID	0.08	0.14	0.24	-	0.29	0.22	0.24	0.31	0.33	0.31	0.32
Psych	0.22	0.19	0.08	0.29	-	0.09	0.08	-	0.07	-	0.06
MB	0.19	0.12	-	0.22	0.09	-	-	0.09	0.14	0.09	0.13
Stan	0.14	0.12	-	0.24	0.08	-	-	0.09	0.13	0.11	0.13
Base	0.21	0.21	0.09	0.31	-	0.09	0.09	-	0.08	-	-
B+MB	0.26	0.24	0.13	0.33	0.07	0.14	0.13	0.08	-	0.07	-
B+Stan	0.21	0.21	0.10	0.31	-	0.09	0.11	-	0.07	-	-
Full	0.24	0.23	0.13	0.32	0.06	0.13	0.13	-	-	-	-

Table D.59: OSE_{All} effect size

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	0.0096	6e-10	6.5e-08	3.1e-20	6.2e-12	6.4e-11	9.1e-20	6.1e-31	4.3e-21	2.4e-27
Emb	0.0096	-	8.1e-06	2.2e-16	1.8e-11	0.058	6e-07	2.1e-16	5.1e-23	1.7e-17	1.3e-20
IC	6e-10	8.1e-06	-	5.4e-31	0.0034	0.022	0.17	2.7e-05	2.8e-10	2.9e-07	1.5e-09
ID	6.5e-08	2.2e-16	5.4e-31	-	7.9e-40	1.2e-21	3.1e-33	2.9e-44	1.2e-54	2.3e-46	9.8e-52
Psych	3.1e-20	1.8e-11	0.0034	7.9e-40	-	1.3e-07	0.016	0.2	1.5e-05	0.051	0.00014
MB	6.2e-12	0.058	0.022	1.2e-21	1.3e-07	-	0.0076	4.5e-08	4.4e-16	3.1e-09	1.1e-13
Stan	6.4e-11	6e-07	0.17	3.1e-33	0.016	0.0076	-	0.00023	7e-09	4.3e-06	4.5e-08
Base	9.1e-20	2.1e-16	2.7e-05	2.9e-44	0.2	4.5e-08	0.00023	-	0.00011	0.29	0.01
B+MB	6.1e-31	5.1e-23	2.8e-10	1.2e-54	1.5e-05	4.4e-16	7e-09	0.00011	-	0.0018	0.1
B+Stan	4.3e-21	1.7e-17	2.9e-07	2.3e-46	0.051	3.1e-09	4.3e-06	0.29	0.0018	-	0.031
Full	2.4e-27	1.3e-20	1.5e-09	9.8e-52	0.00014	1.1e-13	4.5e-08	0.01	0.1	0.031	-

Table D.60: $OSE_{Adv-Elem}$ p-values

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	-	**	**	**	**	**	**	**	**	**
Emb	-	-	**	**	**	-	**	**	**	**	**
IC	**	**	-	**	-	-	-	**	**	**	**
ID	**	**	**	-	**	**	**	**	**	**	**
Psych	**	**	-	**	-	**	-	-	**	-	**
MB	**	-	-	**	**	-	-	**	**	**	**
Stan	**	**	-	**	-	-	-	*	**	**	**
Base	**	**	**	**	-	**	*	-	**	-	-
B+MB	**	**	**	**	**	**	**	**	-	-	-
B+Stan	**	**	**	**	-	**	**	-	-	-	-
Full	**	**	**	**	**	**	**	-	-	-	-

Table D.61: $OSE_{Adv-Elem}$ significance

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	-	0.17	0.15	0.25	0.19	0.18	0.25	0.32	0.26	0.30
Emb	-	-	0.12	0.23	0.18	-	0.14	0.23	0.27	0.23	0.26
IC	0.17	0.12	-	0.32	-	-	-	0.12	0.17	0.14	0.17
ID	0.15	0.23	0.32	-	0.36	0.26	0.33	0.38	0.43	0.39	0.42
Psych	0.25	0.18	-	0.36	-	0.15	-	-	0.12	-	0.10
MB	0.19	-	-	0.26	0.15	-	-	0.15	0.22	0.16	0.20
Stan	0.18	0.14	-	0.33	-	-	-	0.10	0.16	0.13	0.15
Base	0.25	0.23	0.12	0.38	-	0.15	0.10	-	0.11	-	-
B+MB	0.32	0.27	0.17	0.43	0.12	0.22	0.16	0.11	-	-	-
B+Stan	0.26	0.23	0.14	0.39	-	0.16	0.13	-	-	-	-
Full	0.30	0.26	0.17	0.42	0.10	0.20	0.15	-	-	-	-

Table D.62: $OSE_{Adv-Elem}$ effect size

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	0.11	6.8e-11	0.0053	1.9e-25	4.4e-19	2.8e-10	7.5e-22	5.2e-34	3.2e-22	1.2e-32
Emb	0.11	-	5.5e-08	5.6e-08	7.6e-21	8.5e-08	8.4e-08	2.4e-23	1.3e-32	1.4e-23	2.1e-30
IC	6.8e-11	5.5e-08	-	7.3e-25	2.5e-05	0.44	0.88	8.1e-06	1.3e-10	4.1e-07	3.3e-11
ID	0.0053	5.6e-08	7.3e-25	-	2.5e-36	9.2e-19	4.2e-24	4.6e-43	1.1e-50	6.1e-42	2.9e-49
Psych	1.9e-25	7.6e-21	2.5e-05	2.5e-36	-	2.7e-06	7.9e-06	0.71	0.00087	0.56	0.0009
MB	4.4e-19	8.5e-08	0.44	9.2e-19	2.7e-06	-	0.54	1.4e-05	2.3e-12	7.5e-06	2.1e-11
Stan	2.8e-10	8.4e-08	0.88	4.2e-24	7.9e-06	0.54	-	2.5e-06	1.5e-11	7.2e-08	3e-12
Base	7.5e-22	2.4e-23	8.1e-06	4.6e-43	0.71	1.4e-05	2.5e-06	-	1.5e-05	0.8	0.002
B+MB	5.2e-34	1.3e-32	1.3e-10	1.1e-50	0.00087	2.3e-12	1.5e-11	1.5e-05	-	0.00023	0.4
B+Stan	3.2e-22	1.4e-23	4.1e-07	6.1e-42	0.56	7.5e-06	7.2e-08	0.8	0.00023	-	0.00046
Full	1.2e-32	2.1e-30	3.3e-11	2.9e-49	0.0009	2.1e-11	3e-12	0.002	0.4	0.00046	-

Table D.63: OSE_{Close} p-values

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	-	**	-	**	**	**	**	**	**	**
Emb	-	-	**	**	**	**	**	**	**	**	**
IC	**	**	-	**	**	-	-	**	**	**	**
ID	-	**	**	-	**	**	**	**	**	**	**
Psych	**	**	**	**	-	**	**	-	*	-	*
MB	**	**	-	**	**	-	-	**	**	**	**
Stan	**	**	-	**	**	-	-	**	**	**	**
Base	**	**	**	**	-	**	**	-	**	-	-
B+MB	**	**	**	**	*	**	**	**	-	*	-
B+Stan	**	**	**	**	-	**	**	-	*	-	*
Full	**	**	**	**	*	**	**	-	-	*	-

Table D.64: OSE_{Close} significance

	Srp	Emb	IC	ID	Psy	MB	Stan	Base	B+MB	B+Stan	Full
Srp	-	-	0.13	-	0.20	0.17	0.12	0.19	0.24	0.19	0.23
Emb	-	-	0.11	0.11	0.18	0.10	0.10	0.19	0.23	0.19	0.22
IC	0.13	0.11	-	0.20	0.08	-	-	0.09	0.12	0.10	0.13
ID	-	0.11	0.20	-	0.24	0.17	0.20	0.27	0.29	0.26	0.29
Psych	0.20	0.18	0.08	0.24	-	0.09	0.09	-	0.06	-	0.06
MB	0.17	0.10	-	0.17	0.09	-	-	0.08	0.14	0.09	0.13
Stan	0.12	0.10	-	0.20	0.09	-	-	0.09	0.13	0.10	0.14
Base	0.19	0.19	0.09	0.27	-	0.08	0.09	-	0.08	-	-
B+MB	0.24	0.23	0.12	0.29	0.06	0.14	0.13	0.08	-	0.07	-
B+Stan	0.19	0.19	0.10	0.26	-	0.09	0.10	-	0.07	-	0.07
Full	0.23	0.22	0.13	0.29	0.06	0.13	0.14	-	-	0.07	-

Table D.65: OSE_{Close} effect size

D.2.2.2 z-score Normalized difference features

Model	ESEW	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Surprisal	61.95	62.37	67.63	60.49
Embedding	66.18	68.05	74.19	65.25
IntegrationCost	68.01	71.48	76.99	68.42
IdeaDensity	59.79	56.50	57.44	55.55
Psycholinguistic	71.25	76.03	82.35	73.74
ModelBlocks	68.31	73.49	79.78	69.78
Stanford	69.28	70.90	76.98	67.86
Baseline	73.37	78.14	84.69	74.64
Baseline+ModelBlocks	72.91	**78.87**	**85.29**	75.81
Baseline+Stanford	**73.81**	76.93	83.63	73.97
FullModel	73.76	78.32	84.53	**76.04**

Table D.66: Averaged Perceptron performance on **z-score-normalized** difference features using 10-fold cross-validation.

D.2.2.3 Non-normalized difference features with interactions

Model	ESEW	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Surprisal	58.93	63.83	68.98	61.36
Embedding	64.85	65.18	71.55	62.64
IntegrationCost	62.25	64.83	73.36	60.42
IdeaDensity	55.83	50.17	55.77	48.08
Psycholinguistic	66.28	75.15	81.52	72.42
ModelBlocks	61.74	70.07	74.57	67.17
Stanford	62.95	65.44	73.82	61.06
Baseline	72.20	76.71	84.98	72.68
Baseline+ModelBlocks	**72.71**	**79.62**	**86.87**	**76.08**
Baseline+Stanford	70.99	75.40	82.65	71.70
FullModel	72.20	78.57	86.42	75.10

Table D.67: Averaged Perceptron performance on difference features with added **interaction terms** using 10-fold cross-validation.

D.2.2.4 z-score Normalized difference features with interactions

Model	ESEW	OSE_{All}	$OSE_{Adv-Elem}$	OSE_{Close}
Surprisal	62.66	64.48	68.99	61.77
Embedding	66.58	68.58	75.17	65.28
IntegrationCost	66.18	70.22	77.74	67.33
IdeaDensity	60.20	56.78	58.72	56.11
Psycholinguistic	72.40	75.62	86.12	72.04
ModelBlocks	68.97	74.04	81.14	70.08
Stanford	69.22	70.45	78.64	67.25
Baseline	73.79	78.74	85.51	75.06
Baseline+ModelBlocks	73.71	**79.90**	**87.93**	**76.23**
Baseline+Stanford	74.47	77.04	84.98	73.89
FullModel	**75.09**	78.19	**87.93**	75.13

Table D.68: Averaged Perceptron performance on **z-score-normalized** difference features with added **interaction terms** using 10-fold cross-validation.