Smart Selection

Patrick Pantel
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
ppantel@microsoft.com

Michael Gamon
Microsoft Research
One Microsoft Way
Redmond, WA 98052, USA
mgamon@microsoft.com

Ariel Fuxman
Microsoft Research
1065 La Avenida St.
Mountain View, CA 94043, USA
arielf@microsoft.com

Abstract

Natural touch interfaces, common now in devices such as tablets and smartphones, make it cumbersome for users to select text. There is a need for a new text selection paradigm that goes beyond the high acuity selection-by-mouse that we have relied on for decades. In this paper, we introduce such a paradigm, called *Smart Selection*, which aims to recover a user's intended text selection from her touch input. We model the problem using an ensemble learning approach, which leverages multiple linguistic analysis techniques combined with information from a knowledge base and a Web graph. We collect a dataset of true intended user selections and simulated user touches via a large-scale crowdsourcing task, which we release to the academic community. We show that our model effectively addresses the smart selection task and significantly outperforms various baselines and standalone linguistic analysis techniques.

1 Introduction

The process of using a pointing device to select a span of text has a long history dating back to the invention of the mouse. It serves to access functions on text spans, such as copying/pasting, looking up a word in a dictionary, searching the Web, or accessing other accelerators. As consumers move from traditional PCs to mobile devices (e.g., tablets and smartphones), touch interaction is replacing the pointing devices of yore. Although more intuitive and arguably a more natural form of interaction, touch offers much less acuity (colloquially referred to as the *fat finger* problem). To select multi-word spans today, mobile devices require an intricate series of gestures that results in cumbersome user experiences[1]. Consequently, there is an opportunity to reinvent the way users select text in such devices.

Our task is, given a single user touch, to predict the span that the user likely intended to select. We call this task **smart selection**. We restrict our prediction task to cases where a user intends to perform research on a text span (dictionary/thesaurus lookup, translation, searching). We specifically consider operations on text spans that do not form a single unit (i.e., an entity, a concept, a topic, etc.) to be out of scope. For example, full sentences, paragraph and page fragments are out of scope.

Smart selection, as far as we know, is a new research problem. Yet there are many threads of research in the NLP community that identify multi-word sequences, which have coherent properties. For example, named-entity recognizers identify entities such as people/places/organizations, chunkers and parsers identify syntactic constituents such as noun phrases, key phrase detectors or term segmentors identify term boundaries. While each of these techniques retrieve meaningful linguistic units, our problem is a semantic one of recovering a user's *intent*, and as such none alone solves the entire smart selection problem.

In this paper, we model the problem of smart selection using an ensemble learning approach. We leverage various linguistic techniques, such as those discussed above, and augment them with other sources of information from a knowledge

[1]In order to select a multi-word span, a user would first have to touch on either word, then drag the left and right boundary handles to expand it to the adjacent words.

base and a web graph. We evaluate our methods using a novel dataset constructed for our task. We construct our dataset of true user-intended selections by crowdsourcing the task of a user selecting spans of text in a researching task. We obtain 13,681 data points. For each intended selection, we construct test cases for each individual sub-word, simulating the user selecting via touch. The resulting testset consists of 33,912 ⟨simulated selection, intended selection⟩-pairs, which we further stratify into head, torso, and tail subsets. We release the full dataset and testset to the academic community for further research on this new NLP task. Finally, we empirically show that our ensemble model significantly improves upon various baseline systems.

In summary, the major contributions of our research are:

- We introduce a new natural language processing task, called *smart selection*, which aims to address an important problem in text selection for touch-enabled devices;
- We conduct a large crowd-sourced user study to collect a dataset of intended selections and simulated user selections, which we release to the academic community;
- We propose a machine-learned ensemble model for smart selection, which combines various linguistic annotation methods with information from a large knowledge base and web graph;
- We empirically show that our model can effectively address the smart selection task.

2 Related Work

Related work falls into three broad categories: linguistic unit detection, human computer interaction (HCI), and intent detection.

2.1 Linguistic Unit Detection

Smart selection is closely related to the detection of syntactic and semantic units: user selections are often entities, noun phrases, or concepts. A first approach to solving smart selection is to select an entity, noun phrase, or concept that subsumes the user selection. However, no single approach alone can cover the entire smart selection problem. For example, consider an approach that uses a state-of-the-art named-entity recognizer (NER) (Chinchor, 1998; Tjong Kim Sang and De Meulder, 2003; Finkel et al., 2005; Ratinov and Roth, 2009). We found in our dataset (see Section 3.2) that only a quarter of what users intend to select consists in fact of named entities. Although an NER approach can be very useful, it is certainly not sufficient. The remainder of the data can be partially addressed with noun phrase (NP) detectors (Abney, 1991; Ramshaw and Marcus, 1995; Muñoz et al., 1999; Kudo and Matsumoto, 2001) and lists of items in a knowledge base (KB), but again, each is not alone sufficient. NP detectors and KB-based methods are further very susceptible to the generation of false positives (i.e., text contains many nested noun phrases and knowledge base items include highly ambiguous terms).

In our work, we leverage all three techniques in order to benefit from their complementary coverage of user selections. We further create a novel unit detector, called the hyperlink intent model. Based on the assumption that Wikipedia anchor texts are similar in nature to what users would select in a researching task, it models the problem of recovering Wikipedia anchor texts from partial selections.

2.2 Human Computer Interaction

There is a substantial amount of research in the HCI community on how to facilitate interaction of a user with touch and speech enabled devices. To give but a few examples of trends in this field, Gunawardana et al. (2010) address the fat finger problem in the use of soft keyboards on mobile devices, Kumar et al. (2012) explore a novel speech interaction paradigm for text entry, and Sakamoto et al. (2013) introduce a technique that combines touch and voice input on a mobile device for improved navigation of user interface elements such as commands and controls. To the best of our knowledge, however, the problem of smart selection as we defined it has not been addressed.

2.3 Intent detection

There is a long line of research in the web literature on understanding user intent. The closest to smart selection is query recommendation (Baeza-Yates et al., 2005; Zhang and Nasraoui, 2006; Boldi et al., 2008), where the goal is to suggest queries that may be related to a user's intent. Query recommendation techniques are based either on clustering queries by their co-clicked URL patterns (Baeza-Yates et al., 2005) or on leveraging co-occurrences of sequential queries in web search sessions (Zhang and Nasraoui, 2006; Boldi et al., 2008; Sadikov et al., 2010). The key difference from smart selection is that in our task the output is a selection that is relevant to the context of the document where the original selection appears (e.g., by adding terms neighboring the selection). In query recommendation, however, there is no notion of a document being read by the user and, instead, the recommendations are based exclusively on the aggregation of behavior of multiple users.

3 Problem Setting and Data

3.1 Smart Selection Definition

Let $\mathbf{D}$ be the set of all documents. We define a *selection* to be a character $\langle$offset, length$\rangle$-tuple in a document $d \in \mathbf{D}$. Let $\mathbf{S}$ be the set of all possible selections in $\mathbf{D}$ and let $\mathbf{S}_d$ be the set of all possible selections in d.

We define a scored smart selection, σ, in a document d, as a pair $\sigma = \langle x, y \rangle$ where $x \in \mathbf{S}_d$ is a selection and $y \in \mathbb{R}^+$ is a score for the selection.

We formally define the smart selection function ϕ as producing a ranked scored list of all possible selections from a document and user selection pair [2]:

$$\phi : \mathbf{D} \times \mathbf{S} \rightarrow (\sigma_1, ..., \sigma_{|\mathbf{S_d}|} \mid x_i \in \mathbf{S}_d, y_i \geq y_{i+1}) \tag{1}$$

Consider a user who selects s in a document d. Let τ be the target selection that best captures what the user intended to select. We define the smart selection task as recovering τ given the pair $\langle d, s \rangle$. Our problem then is to learn a function ϕ that best recovers the target selection from any user selection.

Note that even for a human, reconstructing an intended selection from a single word selection is not trivial. While there are some fairly clear cut cases such as expanding the selection "Obama" to `Barack Obama` in the sentence "*While in DC, Barack Obama met with...*", there are cases where the user intention depends on extrinsic factors such as the user's interests. For example, in a phrase "*University of California at Santa Cruz*" with a selection "California", some (albeit probably few) users may indeed be interested in the state of `California`, others in the `University of California` system of universities, and yet others specifically in the `University of California at Santa Cruz`. In the next section, we describe how we obtained a dataset of true intended user selections.

[2]The output consists of a ranked list of selections instead of a single selection to allow experiences such as proposing an n-best list to the user.

3.2 Data

In order to obtain a representative dataset for the smart selection task, we focus on a real-world application of users interacting with a touch-enabled e-reader device. In this application, a user is reading a book and chooses phrases for which she would like to get information from resources such as a dictionary, Wikipedia, or web search. Yet, because of the touch interface, she may only touch on a single word.

3.2.1 Crowdsourced Intended Selections

We obtain the intended selections through the following crowdsourcing exercise. We use the entire collection of textbooks in English from Wikibooks[3], a repository of publicly available textbooks. The corpus consists of 2,696 textbooks that span a large variety of categories such as Computing, Humanities, Science, etc. We first produce a uniform random sample of 100 books, and then sample one paragraph from each book. The resulting set of 100 paragraphs is then sent to the crowdsourcing system. Each paragraph is evaluated by 100 judges, using a pool of 152 judges. For each paragraph, we request the judges to select complete phrases for which they would like to "learn more in resources such as Wikipedia, search engines and dictionaries", i.e., our true user intended selections. As a result of this exercise, we obtain 13,681 judgments, corresponding to 4,067 unique intended selections. The distribution of number of unique judges who selected each unique intended selection, in a log-log scale, is shown in Figure 1. Notice that this is a Zipfian distribution since it follows a linear trend in the log-log scale.

Intuitively, the likelihood that a phrase is of interest to a user correlates with the number of judges who select that phrase. We thus use the number of judges who selected each phrase as a proxy for the likelihood that the phrase will be chosen by users.

The resulting dataset consists of 4,067 $\langle d, \tau \rangle$-pairs where d is a Wikibook document paragraph and τ is an intended selection, along with the number of judges who selected it. We further assigned

[3]Available at http://wikibooks.org.

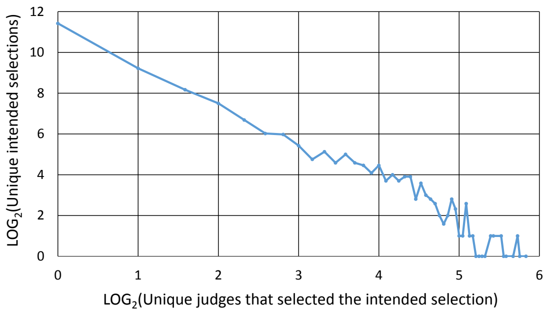


Figure 1: Zipfian distribution of unique intended selections vs. the number of judges who selected them, in log-log scale.

each pair to one of five randomly chosen folds, which are used for cross-validation experiments.

3.2.2 Testset Construction

We define a test case as a triple $\langle d, s, \tau \rangle$ where s is a simulated user selection. For each $\langle d, \tau \rangle$-pair in our dataset we construct n corresponding test cases by simulating the user selections $\{\langle d, \tau, s_1 \rangle, \ldots, \langle d, \tau, s_n \rangle\}$ where $s_1, \ldots, s_n$ correspond to the individual words in τ. In other words, each word in τ is considered as a candidate user selection.

We discard all target selections that only a single judge annotated since we observed that these mostly contained errors and noise, such as full sentences or nonsensical long sentence fragments.

Our first testset, labeled $\mathbf{T}_{\mathtt{ALL}}$, is the resulting traffic-weighted multiset. That is, each test case $\langle d, s, \tau \rangle$ appears k times, where k is the number of judges who selected τ in d. $\mathbf{T}_{\mathtt{ALL}}$ consists of 33,913 test cases.

We further utilize the distribution of judgments in the creation of three other testsets. Following the stratified sampling methodology commonly employed in the IR community, we construct testsets for the frequently, less frequently, and rarely annotated intended selections, which we call HEAD, TORSO, and TAIL, respectively. We obtain these testsets by first sorting each unique selection according to their frequency of occurrence, and then partitioning the set so that HEAD corresponds to the elements at the top of the list that account for 20% of the judgments; TAIL corresponds to the elements at the bottom also accounting for 20% of the judgments; and TORSO corresponds to the remaining elements. The resulting test sets, $\mathbf{T}_{\mathtt{HEAD}}$, $\mathbf{T}_{\mathtt{TORSO}}$, $\mathbf{T}_{\mathtt{TAIL}}$ consist of 114, 2115, and 5798 test cases, respectively[4].

Test sets along with fold assignments and annotation guidelines are available at http://research.microsoft.com/en-us/downloads/eb42522c-068e-404c-b63f-cf632bd27344/.

3.3 Discussion

Our focus on single word selections is motivated by the touchscreen scenario presented in Section 1. Although our touch simulation assumes that each word in a target selection is equally likely to be selected by a user, in fact we expect this distribution to be non-uniform. For example, users may tend to select the first or last word more frequently than words in the middle of the target selection. Or perhaps users tend to select nouns and verbs more frequently than function words. We consider this out of scope for our paper, but view it as an important avenue of future investigation. Finally, for non-touchscreen environments, such as the desktop case, it would also be interesting to study the problem on multi-word user selections.

To get an idea of the kind of intended selections that comprise our dataset, we broke them down according to whether they referred to named entities or not. Perhaps surprisingly, the fraction of named entities in the dataset is quite low, 24.3%[5]. The rest of the intended selections mostly correspond to concepts and topics such as *embouchure formation*, *vocal fold relaxation*, *NHS voucher values*, *time-domain graphs*, etc.

4 Model

As argued in Section 1, existing techniques, such as NER taggers, chunkers, Knowledge Base lookup, etc., are geared towards *aspects* of the task (i.e., NEs, concepts, KB entries), but not the task as a whole. We can, however, combine the outputs of these systems with a learned "meta-model". The meta-model ranks the combined candidates according to a criterion that is derived from data that resembles real usage of smart selection as closely as possible. This technique is known

[4]We stress that $\mathbf{T}_{\mathtt{ALL}}$ is a multi-set, reflecting the overall expected user traffic from our 100 judges per paragraph. $\mathbf{T}_{\mathtt{HEAD}}$, $\mathbf{T}_{\mathtt{TORSO}}$, $\mathbf{T}_{\mathtt{TAIL}}$, in contrast, are not multi-sets since judgment frequency is already accounted for in the stratification process, as commonly done in the IR community.

[5]Becker et al. (2012) report a similar finding, showing that only 26% of questions, which a user might ask after reading a Wikipedia article, are focused on named entities.

in the machine learning community as *ensemble learning* (Dietterich, 1997).

Our ensemble approach, described in this section, serves as our main implementation of the smart selection function ϕ of Equation 1. Each of the ensemble members are themselves a separate implementation of ϕ and will be used as a point of comparison in our experiments. Below, we describe the ensemble members before turning to the ensemble learner.

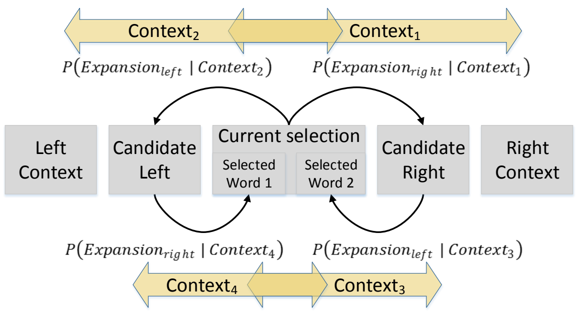
4.1 Ensemble Members

4.1.1 Hyperlink Intent Model

The Hyperlink Intent Model (HIM), which leverages web graph information, is a machine-learned system based on the intuition that anchor texts in Wikipedia are good representations of what users might want to learn about. We build upon the fact that Wikipedia editors write anchor texts for entities, concepts, and things of potential interest for follow-up to other content. HIM learns to recover anchor texts from their single word subselections.

Specifically, HIM iteratively decides whether to expand the current selection (initially a single word) one word to the left or right via greedy binary decisions, until a stopping condition is met. At each step, two binary classifiers are consulted. The first one scores the left expansion decision and the second one scores the right expansion decision. In addition, we use the same two classifiers to evaluate the expansion decision "from the outside in", i.e., from the word next to the current selection (left and right, respectively) to the closest word in the current selection. If the probability for expansion of any model exceeds a predefined threshold, then the most probable expansion is chosen and we continue the iteration with the newly expanded selection as input. The algorithm is illustrated in Figure 2.

We automatically create our training set for HIM by first taking a random sample of 8K Wikipedia anchor texts. We treat each anchor text as an intended selection, and each word in the anchor text as a simulated user selection. For each word to the left (or the right) of the user selection that is part of the anchor text, we create a positive training example. Similarly, for each word to the left (or the right) that is outside of the anchor text, we create a negative training example. We include additional negative examples using random word selections from Wikipedia content. For this purpose we sample random words that are not part of an anchor text. Our final data consists of 2.6M data points, with a 1:20 ratio of positive to negative examples[6].



Context$_2$
Context$_1$
$P(Expansion_{left} \mid Context_2)$
$P(Expansion_{right} \mid Context_1)$
Left Context
Candidate Left
Current selection
Selected Word 1
Selected Word 2
Candidate Right
Right Context
$P(Expansion_{right} \mid Context_4)$
$P(Expansion_{left} \mid Context_3)$
Context$_4$
Context$_3$

Figure 2: Hyperlink Intent Model (HIM) decoding flow for smart selection.

We use logistic regression as the classification algorithm for our binary classifiers. The features used by each model are computed over three strings: the current selection s (initially the single-word simulated user selection), the candidate expansion word w, and one word over from the right or left of s. The features fall into five feature families: (1) *character-level features*, including capitalization, all-cap formatting, character length, presence of opening/closing parentheses, presence and position of digits and non-alphabetic characters, and minimum and average character uni/bi/trigram frequencies (based on frequency tables computed offline from Wikipedia article content); (2) *stopword features*, which indicate the presence of a stop word (from a stop word list); (3) *tf.idf scores* precomputed from Wikipedia content statistics; (4) *knowledge base features*, which indicate whether a string matches an item or a substring of an item in the knowledge base described in Section 4.1.2 below; and (5) *lexical features*, which capture the actual string of the current selection and the candidate expansion word.

4.1.2 Unit Spotting

Our second qualitative class of ensemble members use notions of *unit* that are either based on linguistic constituency or knowledge base presence. The general process is that any unit that subsumes the user selection is treated as a smart selection candidate. Scoring of candidates is by normalized length, under the assumption that in general the most specific (longest) unit is more likely to be the intended selection.

[6]Note that this training set is generated automatically and is, by design, of a different nature than the manually labeled data we use to train and test the ensemble model.

Our first unit spotter, labeled NER is geared towards recognizing named entities. We use a commercial and proprietary state-of-the-art NER system, trained using the perceptron algorithm (Collins, 2002) over more than a million hand-annotated labels.

Our second approach uses purely syntactic information and treats noun phrases as units. We label this model as NP. For this purpose we parse the sentence containing the user selection with a syntactic parser following (Ratnaparkhi, 1999). We then treat every noun phrase that subsumes the user selection as a candidate smart selection.

Finally, our third unit spotter, labeled KB, is based on the assumption that concepts and other entries in a knowledge base are, by nature, things that can be of interest to people. For our knowledge base lookup, we use a proprietary graph consisting of knowledge from Wikipedia, Freebase, and paid feeds from various providers from domains such as entertainment, local, and finance.

4.1.3 Heuristics

Our third family of ensemble members implements simple heuristics, which tend to be high precision especially in the HEAD of our data.

The first heuristic, representing the current touch-enabled selection paradigm seen in many of today's tablets and smartphones, is labeled CUR. It simply assumes that the intended selection is always the user-selected word.

The second is a capitalization-based heuristic (CAP), which simply expands every selected capitalized word selection to the longest uninterrupted sequence of capitalized words.

4.2 Ensemble Learning

In this section, we describe how we train our meta-learner, labeled ENS, which takes as input the candidate lists produced by the ensemble members from Section 4.1, and scores each candidate, producing a final scored ranked list.

We use logistic regression as a classification algorithm to address this task. Our 22 features in ENS consist of three main classes: (1) features related to the individual ensemble members; (2) features related to the user selection; and (3) features related to the candidate smart selection. For (1), the features consist of whether a particular ensemble member generated the candidate smart selection and its score for that candidate. If the candidate smart selection is not in the candidate list of an ensemble member, its score is set to zero. For both (2) and (3), features account for length and capitalization properties of the user selection and the candidate smart selection (e.g., token length, ratio of capitalized tokens, ratio of capitalized characters, whether or not the first and last tokens are capitalized.)

Although training data for the HIM model was automatically generated from Wikipedia, for ENS we desire training data that reflects the true expected user experience. For this, we use five-fold cross-validation over our data collection described in Section 3.2. That is, to decode a fold with our meta-learner, we train ENS with the other four folds. Note that every candidate selection for a $\langle$document, user selection$\rangle$-pair, $\langle d, s\rangle$, for the same d and s, are assigned to a single fold, hence the training process does not see any user selection from the test set.

5 Experimental Results

5.1 Experimental Setup

Recall our testsets $\mathbf{T}_{\texttt{ALL}}$, $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$ from Section 3.2.2, where a test case is defined as a triple $\langle d, s, \tau\rangle$, and where d is a document, s is a user selection, and τ is the intended user selection. In this section, we describe our evaluation metric and summarize the system configurations that we evaluate.

5.1.1 Metric

In our evaluation, we apply the smart selection function $\phi(d, s)$ (see Eq. 1) to each test case and measure how well it recovers τ.

Let $\mathbf{A}$ be the set of $\langle d, \tau\rangle$-pairs from our dataset described in Section 3.2.1 that corresponds to a testset $\mathbf{T}$. Let $\mathbf{T}_{\langle d,\tau\rangle}$ be the set of all test cases in $\mathbf{T}$ with a fixed d and τ. We define the macro precision of a smart selection function, P_ϕ, as follows:

$$P_\phi = \frac{1}{\mid \mathbf{A} \mid} \sum_{\langle d,\tau\rangle \in \mathbf{A}} P_\phi(d, \tau) \tag{2}$$

$$P_\phi(d, \tau) = \frac{1}{\mid \mathbf{T}_{\langle d,\tau\rangle} \mid} \sum_{\langle d,s,\tau\rangle \in \mathbf{T}_{\langle d,\tau\rangle}} P_\phi(d, s, \tau)$$

$$P_\phi(d, s, \tau) = \frac{1}{\mid \phi(d, s) \mid} \sum_{\sigma \in \phi(d,s)} I(\sigma, \tau)$$

$$I(\sigma, \tau) = \begin{cases} 1 & \text{if } \sigma = \langle x, y\rangle \wedge x = \tau \\ 0 & \text{otherwise} \end{cases}$$

	CP@1	CP@2	CP@3	CP@4	CP@5
CUR	39.3	-	-	-	-
CAP	48.9	51.0	51.2	51.8	51.8
NER	43.5	-	-	-	-
NP	34.1	50.2	55.5	57.1	57.6
KB	50.2	50.8	50.9	50.9	50.9
HIM	48.1	48.8	48.8	48.8	48.8
ENS	56.8†	76.0‡	82.6‡	85.2‡	86.6‡

Table 1: Smart selection performance, as a function of CP, on $\mathbf{T}_{\texttt{ALL}}$. ‡ and † indicate statistical significance with $p = 0.01$ and 0.05, respectively. An oracle ensemble would achieve an upper bound CP of 87.3%.

We report cumulative macro precision at rank (**CP@k**) in our experiments since our testsets contain a single true user-intended selection for each test case[7]. However, this is an overly conservative metric since in many cases an alternative smart selection might equally please the user. For example, if our testset contains a user intended selection τ = `The University of Southern California`, then given the simulated selection "California", both τ and `University of Southern California` would most likely equally satisfy the user intent (whereas the latter would be considered incorrect in our evaluation). In fact, the ideal testset would further evaluate the *distance* or *relevance* of the smart selection to the intended user selection. We would then find perhaps that `Southern California` is a more reasonable smart selection than `of Southern California`. However, precisely defining such a *relevance* function and designing the guidelines for a user study is non-trivial and left for future work.

5.1.2 Systems

In our experiments, we evaluate the following systems, each described in detail in Section 4: Passthrough (CUR), Capitalization (CAP), Named-Entity Recognizer (NER), Noun Phrase (NP), Knowledge Base (KB), Hyperlink Intent Model (HIM), Ensemble (ENS).

5.2 Results

Table 1 reports the smart selection performance on the full traffic weighted testset $\mathbf{T}_{\texttt{ALL}}$, as a function of CP@$k$. Our ensemble approach recovers the true user-intended selection in 56.8% of the cases. In its top-2 and top-3 ranked smart selections, the true user-intended selection is retrieved 76.0% and 82.6% of the time, respectively. In position 1, ENS significantly outperforms all other systems with 95% confidence. Moreover, we notice that the divergence between ENS and the other systems greatly increases for $K \geq 2$, where the significance is now at the 99% level.

The CUR system models the selection paradigm of today's consumer touch-enabled devices (i.e., it assumes that the intented selection is always the touched word). Without changing the user interface, we report a 45% improvement in predicting what the user intended to select over this baseline. If we changed the user interface to allow two or three options to be displayed to the user, then we would improve by 93% and 110%, respectively.

For CUR and NER, we report results only at $K = 1$ since these systems only ever return a single smart selection. Note also that when no named entity is found by NER, or no noun phrase is found by NP or no knowledge base entry is found by KB, the corresponding systems return the original user selection as their smart selection.

CAP does not vary much across K: when the intended selection is a capitalized multi-word, the longest string tends to be the intended selection. The same holds for KB.

Whereas Table 1 reports the aggregate expected traffic performance, we further explore the performance against the stratified $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$ testsets. The results are summarized in Table 2. As outlined in Section 3.2, the HEAD selections tend to be disproportionately entities and capitalized terms when compared to the TORSO and TAIL. Hence CAP, NER and KB perform much better on the HEAD. In fact, on the HEAD, CAP performs statistically as well as the ENS model. This means that at position 1, for systems that need to focus only on the HEAD, a very simple solution is adequate. For TORSO and TAIL, however, ENS performs better. At positions 2 and 3, across all strata, the ENS model significantly outperforms all other systems (with 99% confidence).

Next, we studied the relative contribution of each ensemble member to the ENS model. Figure 3 illustrates the results of the ablation study. The ensemble member that results in the biggest performance drop when removed is HIM. Perhaps

[7]Because there is only a single true intended selection for each test case, Recall@k = CP@k.

	HEAD			TORSO			TAIL		
	CP@1	CP@2	CP@3	CP@1	CP@2	CP@3	CP@1	CP@2	CP@3
CUR	48.5	-	-	36.7	-	-	26.6	-	-
CAP	74.2	74.7	74.8	43.0	45.0	45.1	26.1	27.4	28.2
NER	60.6	-	-	39.2	-	-	26.7	-	-
NP	52.3	64.9	69.4	31.0	48.2	53.8	20.0	32.2	35.7
KB	66.7	66.7	66.7	47.0	47.9	48.1	29.9	30.1	30.1
HIM	64.4	65.7	65.7	44.7	45.2	45.4	27.9	28.2	28.2
ENS	75.8	91.8‡	96.5‡	52.7†	73.7‡	81.5‡	32.4†	50.7‡	58.5‡

Table 2: Smart selection performance, as a function of CP, on the $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$ testsets. ‡ and † indicate statistical significance with $p = 0.01$ and 0.05, respectively. An oracle ensemble would achieve an upper bound CP of 98.5%, 86.8% and 64.8% for $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$, respectively.

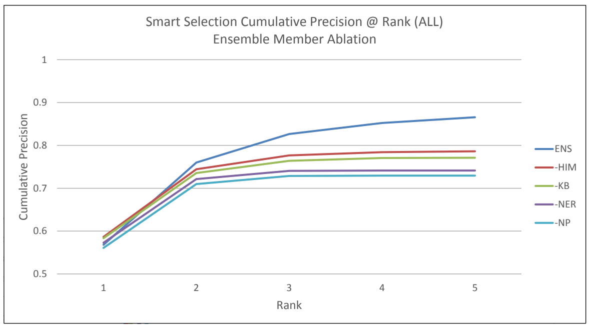


Figure 3: Ablation of ensemble model members over $\mathbf{T}_{\texttt{ALL}}$. Each consecutive model removes one member specified in the series name.

surprisingly, a first ablation of either the CAP or KB model, two of the better individual performing models from Table 1, leads to an ablated-ENS performance that is nearly identical to the full ENS model. One possible reason is that both tend to generate similar candidates (i.e., many entities in our KB are capitalized). Although the HIM model as a standalone system does not outperform simple linguistic unit selection models, it appears to be the most important contributor to the overall ensemble.

5.3 Error Analysis: Oracle Ensemble

We begin by assessing an upper bound for our ensemble, i.e., an *oracle ensemble*, by assuming that if a correct candidate is generated by any ensemble member, the oracle ensemble model places it in first position. For $\mathbf{T}_{\texttt{ALL}}$ the oracle performance is 87.3%. In other words, our choice of ensemble members was able to recover a correct smart selection as a candidate in 87.3% of the user study cases. For $\mathbf{T}_{\texttt{HEAD}}$, $\mathbf{T}_{\texttt{TORSO}}$, and $\mathbf{T}_{\texttt{TAIL}}$, the oracle performance is 98.5%, 86.8%, and 64.8%, respectively.

Although our ENS model's CP@3 is within 2-6 points of the oracle, there is room to significantly improve our CP@1, see Table 1 and Table 2. We analyze this opportunity by inspecting a random sample of 200 test cases where ENS produced an incorrect smart selection in position 1. The breakdown of these cases is: 1 case from $\mathbf{T}_{\texttt{HEAD}}$; 50 cases from $\mathbf{T}_{\texttt{TORSO}}$; 149 cases from $\mathbf{T}_{\texttt{TAIL}}$, i.e., most errors occur in the TAIL.

For 146 of these cases (73%), not a single ensemble member produced the correct target selection τ as a candidate. We analyze these cases in detail in Section 5.4. Of the remaining cases, 25, 10, 9, 4, 4, and 2 were correct in positions 2, 3, 4, 5, 6, 7, respectively. Table 3 lists some examples.

In 18 cases (33%), the result in position 1 is very reasonable given the context and user selection (see lines 1-4 in Table 3 for examples). Often the target selection was also found in second position. These cases highlight the need for a more relaxed, relevance-based user study, as pointed out at the end of Section 5.1.1.

We attributed 7 (13%) of the cases to data problems: some cases had a punctuation as a sole character user selection, some had a mishandled escaped quotation character, and some had a UTF-8 encoding error.

The remaining 29 (54%) were truly model errors. Some examples are shown in lines 5-8 in Table 3. We found three categories of errors here. First, our model has learned a strong prior on preferring the original user selection (see example line 5). From a user experience point of view, when the model is unsure of itself, it is in fact better not to alter her selection. Second, we also learned a strong capitalization prior, i.e., to trust the CAP member (see example line 6). Finally, we noticed that we have difficulty handling user selections consisting of a stopword (we noted determiners, prepositions, and the word "and"). Adding a few simple features to ENS based on a stopwords list or a list of closed-class words should address this problem.

	Text Snippet	User Selection	`ENS` 1st Result
1	"The **Russian conquest of the South Caucasus in the 19th century** split the speech community across two states..."	Caucasus	`South Caucasus`
2	"...are generally something that **transportation** agencies would like to minimize..."	transportation	`transportation agencies`
3	"The **vocal organ of birds**, the syrinx, is located at the base of the blackbird's trachea."	vocal	`vocal organ`
4	"An example of this may be an idealised **waveform** like a square wave..."	waveform	`idealised waveform`
5	"Tickets may be purchased from either the **ticket counter** or from automatic machines..."	counter	`counter`
6	"PBXT features include the following: **MVCC** Support: MVCC stands for Multi-version Concurrency Control."	MVCC	`MVCC Support`
7	"Centers for song production pathways include the High vocal center; **robust nucleus of archistriatum** (RA); and the tracheosyringeal part of the hypoglossal nucleus..."	robust	`robust nucleus`
8	"...and get an **11gR2 RAC cluster database** running inside virtual machines..."	cluster	`RAC cluster`

Table 3: Position 1 errors when applying `ENS` to our test cases. The text snippet is a substring of a paragraph presented to our judges with the target selection (τ) indicated in bold.

5.4 Error Analysis: Ensemble Members

Over all test cases, the distribution of cases without a correct candidate generated by an ensemble member in the `HEAD`, `TORSO`, `TAIL` is 0.3%, 34.6%, and 65.1%, respectively. We manually inspected a random sample of 100 such test cases.

The majority of them, 83%, were large sentence fragments, which we consider out of scope according to our prediction task definition outlined in Section 1. The average token length of the target selection τ for these was 15.3. In comparison, we estimate the average token length of the task-admissable cases to be 2.7 tokens. Although most of these long fragment selections seem to be noise, a few cases are statements that a user would reasonably want to know more about, such as: (i) "Talks of a merger between the NHL and the WHA were growing" or (ii) "NaN + NaN * 1.0i".

In 10% of the cases, we face a punctuation-handling issue, and in each case our ensemble was able to generate a correct candidate when fixing the punctuation. For example, for the book title τ = `What is life?`, our ensemble found the candidate `What is life`, dropping the question mark. For τ = `Near Earth Asteroid.` our ensemble found `Near Earth Asteroid`, dropping the period. Similar problems occurred with parentheses and quotation marks.

In two cases, our ensemble members dropped a leading "the" token, e.g., for τ = `the Hume Highway`, we found `Hume Highway`.

Finally, 2 cases were UTF-8 encoding mistakes, leaving five "true error" cases.

6 Conclusion and Future Work

We introduced a new paradigm, *smart selection*, to address the cumbersome text selection capabilities of today's touch-enabled mobile devices. We report 45% improvement in predicting what the user intended to select over current touch-enabled consumer platforms, such as iOS, Android and Windows. We release to the community a dataset of 33, 912 crowdsourced true intended user selections and corresponding simulated user touches.

There are many avenues for future work, including understanding the distribution of user touches on their intended selection, other interesting scenarios (e.g., going beyond the e-reader towards document editors and web browsers may show different distributions in what users select), leveraging other sources of signal such as a user's profile, her interests and her local session context, and exploring user interfaces that leverage n-best smart selection prediction lists, for example by providing selection options to the user after her touch.

With the release of our 33, 912-crowdsourced dataset and our model analyses, it is our hope that the research community can help accelerate the progress towards reinventing the way text selection occurs today, the initial steps for which we have taken in this paper.

7 Acknowledgments

The authors thank Aitao Chen for sharing his NER tagger for our experiments, and Bernhard Kohlmeier, Pradeep Chilakamarri, Ashok Chandra, David Hamilton, and Bo Zhao for their guidance and valuable discussions.

References

Steven. P. Abney. 1991. Parsing by chunks. In Robert C. Berwick, Steven P. Abney, and Carol Tenny, editors, *Principle-Based Parsing: Computation and Psycholinguistics*, pages 257–278. Kluwer, Dordrecht.

Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2005. Query recommendation using query logs in search engines. In *Current Trends in Database Technology-EDBT 2004 Workshops*, pages 588–596. Springer.

Lee Becker, Sumit Basu, and Lucy Vanderwende. 2012. Mind the gap: Learning to choose gaps for question generation. In *Proceedings of NAACL HLT '12*, pages 742–751.

Paolo Boldi, Francesco Bonchi, Carlos Castillo, Debora Donato, Aristides Gionis, and Sebastiano Vigna. 2008. The query-flow graph: model and applications. In *Proceedings of CIKM '08*, pages 609–618. ACM.

Nancy A. Chinchor. 1998. Named entity task definition. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, Fairfax, VA.

Michael Collins. 2002. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proceedings of EMNLP*.

Thomas G. Dietterich. 1997. Machine Learning Research - Four Current Directions. *AI Magazine*, 18:4:97–136.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *In ACL*, pages 363–370.

Asela Gunawardana, Tim Paek, and Christopher Meek. 2010. Usability guided key-target resizing for soft keyboards. In *Proceedings of IUI '10*, pages 111–118.

Taku Kudo and Yuji Matsumoto. 2001. Chunking with support vector machines. In *Proceedings of NAACL '01*, pages 1–8.

Anuj Kumar, Tim Paek, and Bongshin Lee. 2012. Voice typing: A new speech interaction model for dictation on touchscreen devices. In *Proceedings of CHI'12*, pages 2277–2286.

Marcia Muñoz, Vasin Punyakanok, Dan Roth, and Dav Zimak. 1999. A learning approach to shallow parsing. In *Proceedings of EMNLP/VLC*, pages 168–178.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the 3rd ACL Workshop on Very Large Corpora*, pages 82–94. Cambridge MA, USA.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-2009*, pages 147–155.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Mach. Learn.*, 34(1-3):151–175, February.

Eldar Sadikov, Jayant Madhavan, Lu Wang, and Alon Halevy. 2010. Clustering query refinements by user intent. In *Proceedings of the 19th international conference on World wide web*, pages 841–850. ACM.

Daisuke Sakamoto, Takanori Komatsu, and Takeo Igarashi. 2013. Voice augmented manipulation: using paralinguistic information to manipulate mobile devices. In *Mobile HCI*, pages 69–78.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of CoNLL-2003*, pages 142–147. Edmonton, Canada.

Zhiyong Zhang and Olfa Nasraoui. 2006. Mining search engine query logs for query recommendation. In *Proceedings of the 15th international conference on World Wide Web*, pages 1039–1040. ACM.