STANFORD UNIVERSITY

Categorization by Character-Level Models: Exploiting the Sound Symbolism of Proper Names

A thesis submitted in partial satisfaction of the
requirements for the degree Master of Science

in Symbolic Systems

by

Joseph Robert Smarr

Committee in charge:

Professor Christopher D. Manning, Advisor, Primary Thesis Reader
Professor Pat Langley, Second Thesis Reader

The Masters Thesis of Joseph Robert Smarr is approved, and it is acceptable in quality and form for a master's project in Symbolic Systems.

Primary thesis reader

Secondary thesis reader

TABLE OF CONTENTS

Abstract

This thesis presents a context-, domain-, and language-independent method for classifying proper names into semantic categories. The core observation is that many proper names contain character sequences that are distinctive of their category. There is, in effect, a surprisingly strong sound-symbolic relationship between names and the things they describe, which can be uncovered with machine learning methods. While previous work has exploited a portion of this form-meaning relationship (modeling suffixes, patterns of capitalization or punctuation, etc.), this thesis presents a generalized approach in which character-level subsequences are the basic unit of evidence and every piece of a proper noun can be used as input. The result is a highly general, robust, and accurate classifier that can categorize proper names without surrounding context and straightforwardly handle previously unseen words. We describe a probabilistic generative model based on *n*-gram character sequences and present experiments classifying a wide variety of names that include companies, people, places, pharmaceutical drugs, movie titles, proteins, car models, musical artists, and diseases. We show how this same model can be used to generate new proper names that mimic a given category. We then generalize this model to perform segmentation alongside classification, resulting in a named-entity recognizer that achieves competitive results in both English and German. Finally, we discuss the relationship of this work to sound symbolism, language identification, and professional brand-name creation, investigating the origins of the self-descriptive naming conventions that our model is able to capture.

Acknowledgements

I wish to extend my most sincere appreciation for all the people that have educated, supported, and challenged me throughout my career at Stanford. First, I wish to thank my parents and brother, who have provided me with so many opportunities, and who have always encouraged me and provided perspective. I thank also my dearest beloved girlfriend Michelle, who has patiently and cheerfully stuck by me through many late nights of research and manic bouts of work. Second, I wish to thank my academic peers, most notably Dan Klein, who have spent countless hours debating the details of this research with me (and who pointed out many key insights), and my professors who have taught me and spent considerable time with me outside the classroom, despite my often over-eager personality. In particular, I wish to single out three professors who have had a particularly profound impact on me. Pat Langley took me under his wing as a young student excited by artificial intelligence, and taught me to be passionately dispassionate about following any one school of thought or method du jour. It was a valuable lesson that has lead me to explore a wide variety of approaches to AI problems, and to appreciate the similarities and differences between them. Ivan Sag transformed my interest in language into a passion for linguistics, and specifically for developing precise linguistic theories that accurately describe a broad range of phenomena. It was this perspective that honed my desire to build language-processing computer programs as concrete instantiations of linguistic hypotheses. And, most importantly, Chris Manning acted as my research mentor for two years, echoing my enthusiasm for building practical natural language technology by gaining a deeper understanding of both machine learning and linguistics. Chris gave me supreme flexibility to pursue my interests, and yet he always had valuable insights to share and knowledge of related work to point me to. He funded my coterminal 5th year at Stanford as a research assistant, and despite some early disappointments in getting papers accepted for publication, he never stopped believing in me and always pushed me to do my very best. Finally, I want to thank the Symbolic Systems Program itself for creating such a rare and wonderful framework in which I could pursue my inherently interdisciplinary interests.

Chapter 1:

The challenge of unknown words

A major challenge in building robust, wide-coverage natural language processing (NLP) systems is coping with the unrelenting amount of unknown words. Most methods rely on observing repeated occurrences of data in order to measure correlations between relevant pieces of information (either tacitly for building rules by hand or explicitly for machine learning approaches). For example, statistical part of speech taggers usually predict tags by remembering the most common part of speech for each word observed in a large, manually labeled corpus of word-tag pairs. When an unknown word is encountered, no such information is available, and the system is forced to guess (Weischedel et al. 1993). Thus, unknown words represent a significant source of error for many NLP systems (Bazzi & Glass 2000).

1.1. Quantifying unknown words

To a certain extent, the problem of unknown words can be mitigated by accumulating larger and larger quantities of text (Armstrong et al. 1999, Banko & Brill 2001). However, it has been widely observed that word frequency is a *heavy-tailed* distribution, meaning that an appreciable amount of the probability mass is in the tail of the distribution, and thus new words will continue to show up with considerable frequency even after collecting very large corpora (Chitashvili 1993). The famous Zipf's law states that the frequency of a word (the number of times it occurs in a given amount of text) is inversely proportional to its rank (how common it is) (Zipf 1949). Thus, word frequency follows a power law, a function that is notorious for taking a long time to decay.

Empirically, as the size of a corpus increases, the occurrence of previously unseen words grows at approximately the square-root of the total number of word tokens seen.[1]

[1] A *token* of word is a single occurrence in a document, whereas its *type* refers to the unique word itself. Thus, for example, the single word type *the* has numerous tokens in normal text, whereas a rare word type like *antepenultimate* may have only one (or no) tokens even in a large corpus. In the results presented in this thesis, word types are determined by string identity. For example, *the* and *The* are considered different types because their spelling differs in case. Similarly, *look* and *looked* are different types.

Figure 1 presents a replication of this well-known fact (e.g., Ara'ujo et al. 1997) on sections 0 through 20 of the Wall Street Journal corpus, obtained from the Penn Treebank (Marcus et al. 1993). Even after seeing a million words of text, the occurrence of new words has not yet leveled off. A major cause of the continual introduction of new words is the inherent *non-stationarity* of language. People are constantly talking about new events, new concepts, and new people, so two documents taken from different times—even relatively close together—will reflect different vocabularies and distributions of word frequency.

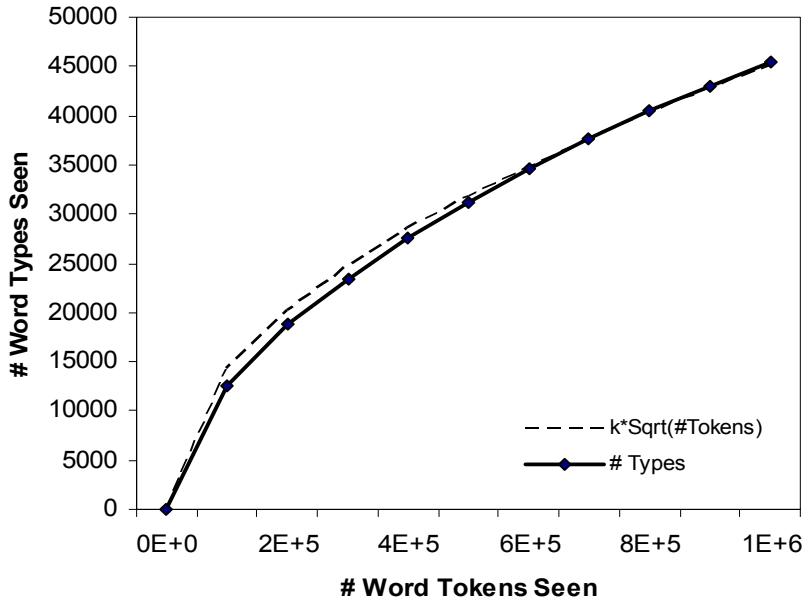

Figure 1. Growth of word types (new words) vs. word tokens (all words).

As more tokens of text are encountered (x-axis), the number of unique word types (y-axis) empirically grows at roughly the square-root (comparison uses *k*=45).

The problem of unknown words is particularly acute when dealing with proper nouns or names. Proper nouns include people's names, as well as the names of companies, products and places. They represent the most open and productive class of words in English—new company and product names are constantly being created, and they rank at or near the top of all syntactic categories in terms of sheer number of types and rate of growth.

Figure 2 shows the relative breakdown of unknown words by part of speech. These numbers were obtained by counting the number of words in sections 22 through 24 of the Wall Street Journal corpus that were never seen in sections 0 through 20. Proper nouns account for over 40% of the unknown tokens encountered and over 30% of the types. The disparity between tokens and types is due to the "burstiness" of names: once a new name is introduced, it is commonly referred to many times.

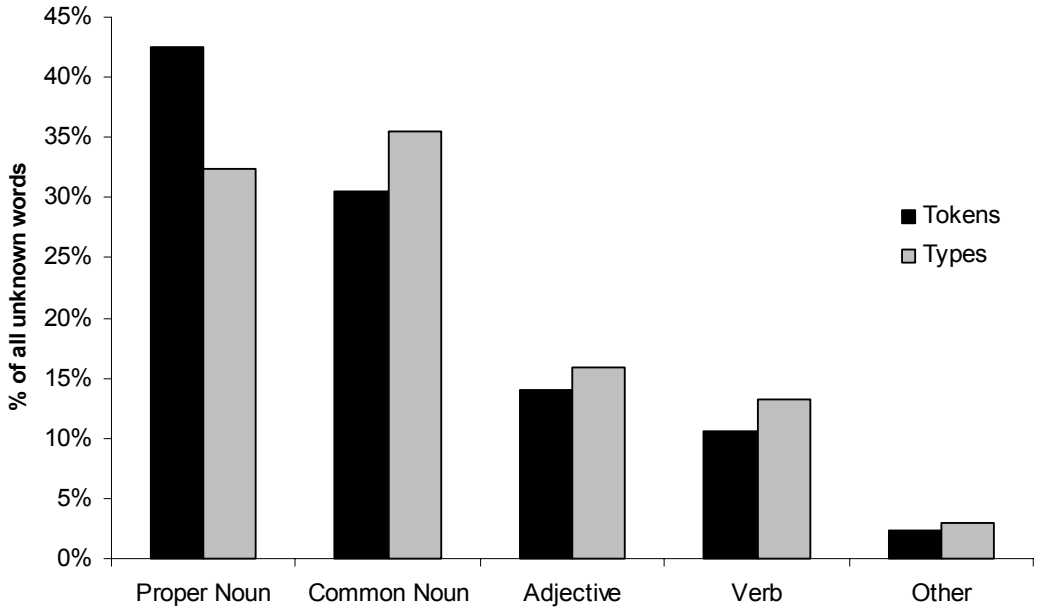

Figure 2. Unknown words by part of speech.

All words in sections 0-20 of WSJ were considered "known" and only words occurring for the first time in sections 22-24 were counted as "unknown". Distinction in number (e.g., NN/NNS) and inflection (e.g., VBZ/VBG) was collapsed.

Not only are unknown proper nouns numerous by quantity; they are also the most productive word class. In other words, as a syntactic class, the rate of occurrence of new proper nouns takes the longest time to decay. Figure 3 shows the proportion of words in each syntactic class that have already been seen. As more text is read, all classes continue to "fill up" their inventory of words, but proper nouns lag furthest behind. Even after reading a million words of text, the chance that the next proper noun encountered will be previously unseen is over 10%.

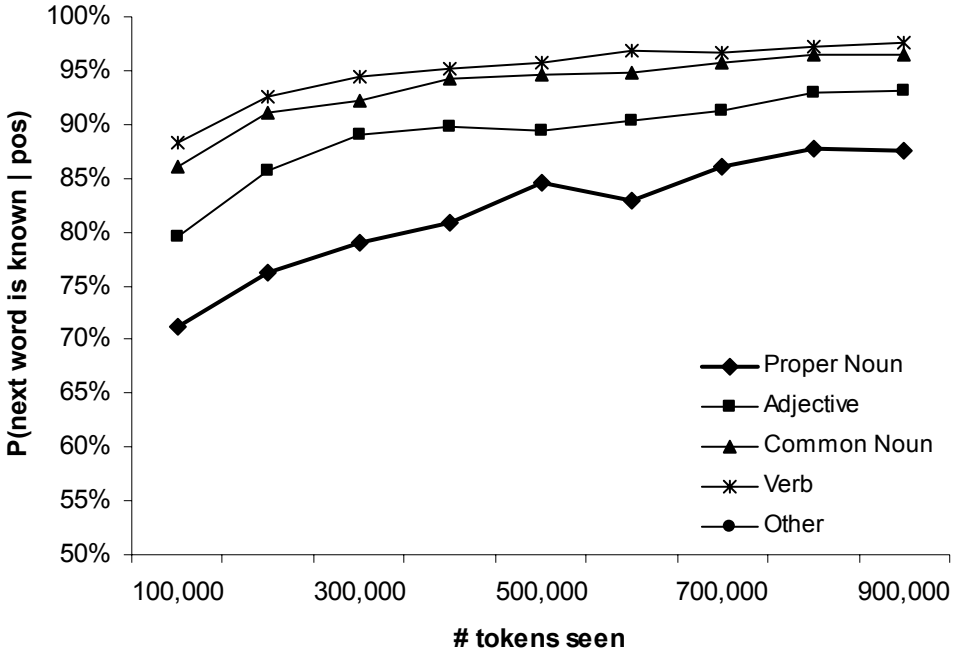

Figure 3. Saturation rates by part of speech.

The x-axis shows the progressive buildup of corpus in 100,000 word chunks, and at each point the fraction of unknown words (words encountered for the first time in the current chunk) are computed for each major part of speech.

1.2. The importance of recognizing proper noun phrases

While proper nouns are the word class most riddled by unknown word problems, they are also one of the most important classes for NLP tasks. Recognizing and classifying proper noun phrases (PNPs) in text is a key challenge in many subfields of NLP, including shallow parsing (also called *chunking*, e.g., Ramshaw & Marucs 1995), named-entity recognition (e.g., Mikheev et al. 1999), and information extraction (e.g., Freitag & McCallum 2000). The two primary subtasks are *segmenting* PNPs (marking their boundaries within continuous text) and *classifying* them into a semantic class (distinguishing different types of names). For example, consider what is required to correctly annotate the two sentences:

The [$_{\text{ORG}}$ **Microsoft**] Quarterly Report, said [$_{\text{PER}}$ **Huy Nguyen**], showed earnings growth.

In [$_{\text{LOC}}$ **Japan**] [$_{\text{ORG}}$ **Sony Electronics**] continues to outperform the broad market.

Among other things, the system needs to recognize that "Microsoft" by itself is an organization (even though it is inside the capitalized block "The Microsoft Quarterly Report", it needs to recognize that "Huy Nguyen" is a person's name (even though it is an uncommon name in English), and it must recognize that "Japan" and "Sony" are two adjacent names (rather than one big name).

A great deal of research in *named-entity recognition* took place as part of the annual U.S. government-sponsored Message Understanding Conferences (MUC), which required identifying PNPs in text and classifying them as company names, place names, and people names (Mikheev et al. 1998, Collins et al. 1999). Similarly, several studies have tried to identify novel terms in medical abstracts (e.g., Campbell et al. 1999, Bodenreider et al. 2000). Recognizing proper names is also of ancillary importance in tasks such as part of speech tagging, parsing, and information retrieval, which depend on having an informative lexicon and thus need a way of productively extending it when dealing with unknown words. The persistence of focus placed on these tasks is largely due to the challenge of robustly identifying PNPs in spite of the high occurrence of unknown words.

1.3. Coping with unknown words

When faced with previously unknown words, there are two sources of information to fall back on—the surrounding *context* in which the word occurs, and/or some distinguishing *content* feature of the word itself. Contextual cues are typically syntactic patterns that prefer a particular class of PNP. For example, in a sentence like "NP_X was appointed President of NP_Y", *NP_X* is most likely a person and *NP_Y* is most likely a company. Similarly, a sentence like "NP_Z was approved by the FDA" might indicate that *NP_Z* is a drug. The two major approaches for generating contextual cues have been hand-built, domain-specific rules based on patterns in the syntactic context (Appelt et al. 1995, Mikheev et al. 1998, Bodenreider et al. 2000), and machine-learning approaches that take labeled data and build up surrounding patterns (Soderland 1995, Riloff 1996, Muslea et al. 1999, Freitag & Kushmerick 2000).

A complementary strategy for using context to classify PNPs is to use internal features of the PNP itself. Traditional approaches to PNP classification rely primarily on

large, manually constructed lists of known names or "gazetteers" (Wacholder et al. 1997, Charoenpornsawat et al. 1998, Mikheev et al. 1998, Bodenreider et al. 2000), and/or gross word-level features such as capitalization, punctuation, or presence of numbers and other "special characters" (Bikel et al. 1997, Wacholder et al. 1997, Baluja et al. 1999, Bikel et al. 1999, Bodenreider et al. 2000). Many of these systems use some form of machine learning in conjunction with these features. These methods achieve relatively high levels of performance, but suffer from the problem that building lists of words or heuristic rules is slow, expensive, and domain specific. These shortcomings have been widely acknowledged (e.g., Wacholder et al. 1997, Mihkeev et al. 1999), yet few alternative strategies for unknown PNP classification have been proposed.

1.4. Can names be classified without context?

It is not immediately obvious through introspection what features humans use to classify previously unseen names. In particular, it is unclear whether context or content plays a more important role. It is common to point to names like *Washington* that can refer either to a person, place, or organization as a demonstration that classifying names without context is hopeless. Even if one considers potentially informative suffix patterns like *–ville* for place names (e.g., *Knoxville*, *Yountville*), there are obvious counter examples, such as the person's name *Neville*. On the other hand, in many cases names can be quite descriptive independent of any context. For example, company names often end in *Inc.* or *Corp*. Thus, even if the main part of a company's name is an unknown word, the name as a whole may be quite identifiable (e.g., *Plaxo, Inc.*). Single-word names can also be quite discernable at times. For example, pharmaceutical drugs such as *Novo-Doxylin* and *Cotramoxizole* are strongly marked and easily recognizable. In these cases, since the word itself is unknown and it occurs without any surrounding context, the only pieces of evidence available for classification are *word-internal* features, such as common character sequences and word length.

The central thesis of this research is that the internal composition of PNPs provides surprisingly strong evidence for classification, independent of any context. Despite the concern over words like *Washington* or character sequences like *–ville*, this work demonstrates that in many cases, character-level features provide pervasive and

reliable cues for categorizing names into semantic classes, even when some or all of the words in the name have never been seen before. In general, identifying instances of proper names in a pre-defined list (whether manually constructed or learned) is relatively easy. The difficulty comes when trying to segment and classify previously unseen names. Although unknown word models of varying sophistication have been developed, in most cases they are an add-on to a core system that is focused elsewhere, and the set of features proposed to distinguish different types of names has been largely ad hoc and language dependent. This thesis treats proper handling of unknown words as a central challenge in NLP, and strives for a more general and principled solution to the problem than is commonly found.

With this goal in mind, the remainder of the thesis is organized as follows: Chapter 2 describes a statistical model that can be used to classify PNPs after being trained on labeled examples. Chapter 3 presents a number of classification experiments with this model that demonste the power and flexibility of character-level features. Chapter 4 analyzes these results in more detail, with particular attention to the types of errors the model makes and the contribution of different model components to overall performance. Chapter 5 builds on the PNP classification model to include segmentation along with classification, and presents results using this combined model for multilingual named-entity recognition. Chapter 6 speculates on the linguistic and cultural forces that give rise to the apparently systematic naming conventions that these models exploit. Finally, Chapter 7 concludes by summarizing the findings of this research and identifying additional contributions that could be made in the future.

Chapter 2:

A probabilistic model of proper noun phrases

Driven by the observation that unknown names can often be classified without surrounding context, this chapter formalizes the task of proper noun phrase (PNP) classification and presents a probabilistic generative model that can be trained on manually classified PNPs and used to classify novel cases. The majority of the exposition and experiments discussed below deal with five categories of PNPs: drug names, company names, movie titles, place names, and people's names. Later we present classification experiments with a number of other categories to demonstrate the generality of the model and to investigate which types of names are easier and harder to categorize.

2.1. Formalization of problem

In this section we formalize the task and procedure of PNP classification and its evaluation. The work presented here is largely based on (Smarr & Manning 2002).

2.1.1. Performance Task

The performance task is to take a string representing a Proper Noun Phrase (e.g., *Aureomycin* or *Keyspan Corp*) and classify it into one of a predefined set of categories (e.g., *drug name* or *company name*). A *Proper Noun Phrase* is a sequence of one or more words that represents the name of a person, place, or thing. As mentioned above, our goal is to assess the ability to classify an already segmented PNP independent of context.

2.1.2. Training

We use a standard supervised learning paradigm. A training example consists of a PNP labeled with its semantic category. A portion of the training examples (20% in the reported results) are held out for learning various parameters described in the next section, and the remainder are counted to derive various statistics. After cross-validation parameters have been set, the held-out data is also trained on before testing.

2.1.3. Testing

After training is completed, the classifier is presented with another list of PNPs, none of which appeared in the training data. Some PNPs are inherently ambiguous and thus could be judged as correctly belonging to multiple classes (e.g., *Washington* is both a place and a name). However, we follow the stringent evaluation standard of only accepting the category from which the example was originally taken, regardless of its ambiguity.

2.1.4. Evaluation

Each result presented is the average of ten separate train/test runs. In each case, a randomly selected 90% of the supervised data is used for training, and the remaining 10% is used for testing. The evaluation metric we use is raw classification accuracy, defined as the number of test examples for which the correct category was provided by the classifier, divided by the total number of test examples presented.

2.2. Model used for classification

Classification of PNPs is performed using a probabilistic generative model for each category. The predicted category of a new PNP is determined by the following formula:

$$\text{Predicted-Category}(pnp) = \text{argmax}_c\ P(c|pnp) = \text{argmax}_c\ P(c)^{\alpha} \times P(pnp|c)$$

where $P(c)$ is estimated empirically from the training data, and α is a prior-boost, set with a line search on held-out data.[2] $P(pnp|c)$ is a model of each category of PNPs that is based on the number and length of words used and the composition of each word.

For an intuition about the empirical value of character sequences and word-length information for PNP classification, consider the three names:

Cotrimoxazole

Wethersfield

Alien Fury: Coundown to Invasion

[2] A "line search" simply means testing several parameter values in a given range and picking the one that resulted in the best score. In practice, setting a prior boost usually made little difference, as it was often set to 1.0 during cross validation.

The first name is clearly a drug name, and this is strongly represented in the character sequence. For example, the trigram *oxa* appears in 18 drug names in the training data, and no names of any other category. Thus the presence of this substring is strongly correlated with the category drug names. Similarly, the substring *field* occurs in 68 place names, 14 people's names, eight company names, and six movie titles. Thus while it is not as strong a cue as *oxa*, it is still clearly informative. Finally, a *colon character* occurs in 708 movie titles, six drug names, and nothing else. Thus, single characters along with short and long character sequences together provide a multitude of category predictors.

Another marked difference between PNP categories is the length and number of words used. Figure 4 shows the empirical distribution of characters per word (left) and words per PNP (right) for several categories of PNPs. For example, a PNP with four words is very unlikely to be a place name, but it may well be a movie title. Words with ten or more characters are likely to be part of drug or place names than company names or movie titles. Like character sequences, most of these cues are not100% effective on their own, but the accumulation of evidence from many such sources produces an overall ranking of likely categories that is quite reliable.

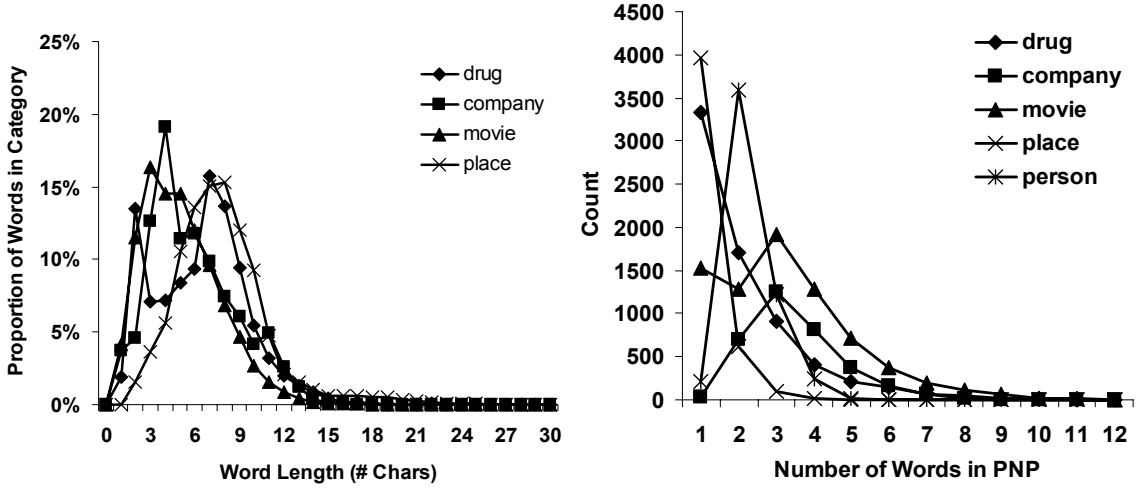

Figure 4. Word-length and number-of-word distributions by PNP category.

The left graph shows the normalized histogram of word lengths for each word in each proper noun phrase (per category). For instance, note the bimodal distribution in drug-name words, reflecting the presence of long (main) words (e.g., *Acetaminophen*) and short (suffix) words (e.g., *DM*).

The right graph shows the raw histogram of number of words per complete proper noun phrase (per category). For instance, most place names are a single word (e.g., *California*) while most movie titles are many words long (e.g., *Natural Born Killers*).

Formally, P(*pnp*|*c*) is a generative model of the form

$$P(pnp|c) = P_{n\text{-gram}}(\textbf{word-lengths}(pnp)) \times \Pi_{\text{word } i \in pnp}\, P(w_i|\textbf{word-length}(w_i))$$
$$P(w_i|len) = \lambda_{len} \times P_{\text{n-gram}}(w_i|len)^{k/len} + (1-\lambda_{len}) \times P_{\text{word}}(w_i|len)$$

where each term is implicitly conditioned on the category. The probability for each word is computed as the weighted average of the character *n*-gram estimate and the known-word estimate. Interpolation weights are learned for each distinct word length using a line search on held-out data.[3] P(*w*|*len*) is estimated empirically as the number of times word *w* was seen in the training data divided by the total number of words of length *len* characters seen.

The function **word-lengths**(*pnp*) returns a list of integers, representing the number of characters in each word of the PNP. The list is prepended with (*n*–1) boundary symbols for conditioning, and a final "zero-length word" indicating the end of the PNP is also added so that the termination of PNPs becomes a statistically observable event. For example, **word-lengths**(*Curtis E. Lemay*) = [6, 2, 5, 0].

Two *n*-gram (Markov) models are used, one for word lengths and one for characters. We set $n = 4$ for lengths and $n = 6$ for characters, and we used deleted interpolation to estimate probabilities, using the following recursive function:

$$P_{0\text{-gram}}(symbol|history) = \text{uniform-distribution}$$
$$P_{\text{n-gram}}(s|h) = \lambda_{C(h)} P_{\text{empirical}}(s|h) + (1 - \lambda_{C(h)}) P_{\text{(n-1)-gram}}(s|h)$$

Thus, the 2-gram estimate is a mix of the empirical 2-gram distribution and the combined 1/0-gram distribution, the 3-gram estimate is a mix of the empirical 3-gram and the combined 2/1/0-gram, and so forth. Interpolation parameters $\lambda_{C(h)}$ are estimated via expectation maximization (EM) on held-out data, and are learned separately based on the binned count of the conditioning context C(*h*). We utilized six bins in our experiments: {0, ≤5, ≤50, ≤500, ≤5000, >5000}, although we found only minor deviation in our results by changing the specific bins used.

[3]If a novel word length is seen during testing, the interpolation parameter is automatically set to 1 for the character n-gram and 0 for the word model, since no words of that length have been seen before.

The character *n*-gram estimate for the entire word is conditioned on word length (to maintain independence with the word-length model) by dividing by the fraction of words in the training data with the given number of characters. The first word in the PNP is prepended with $n - 1$ spaces (starter symbols, with the effect that the 2-gram and lower estimates treat the first word identically to a middle word) and the subsequent words are prepended with the preceding $n - 1$ chars in the PNP (including the preceding space). Each *n*-gram word estimate is terminated after reaching the following space character. A unique termination symbol is appended after the last word in each PNP. The probability of each word is normalized for length by taking the $(k/\text{length})^{\text{th}}$ root, which is discussed in more detail in Section 4.3.2.

As a concrete example of how the model works, consider the classification of the name "Alec Baldwin" (an actor). Figure 5 shows the cumulative score (in log probability) of this PNP for each category as evidence builds up. At first there is simply the prior probability (overall rate) of each category. Next the word-length sequence is considered in three steps (4-7-0), leaving *person* on top but only slightly. Next each word is scored based on its character sequences. At the end, *person* is a clear winner. Note that *place* follows relatively closely because the word *Baldwin* shows up once as a place name (in *Brightwell Baldwin*).

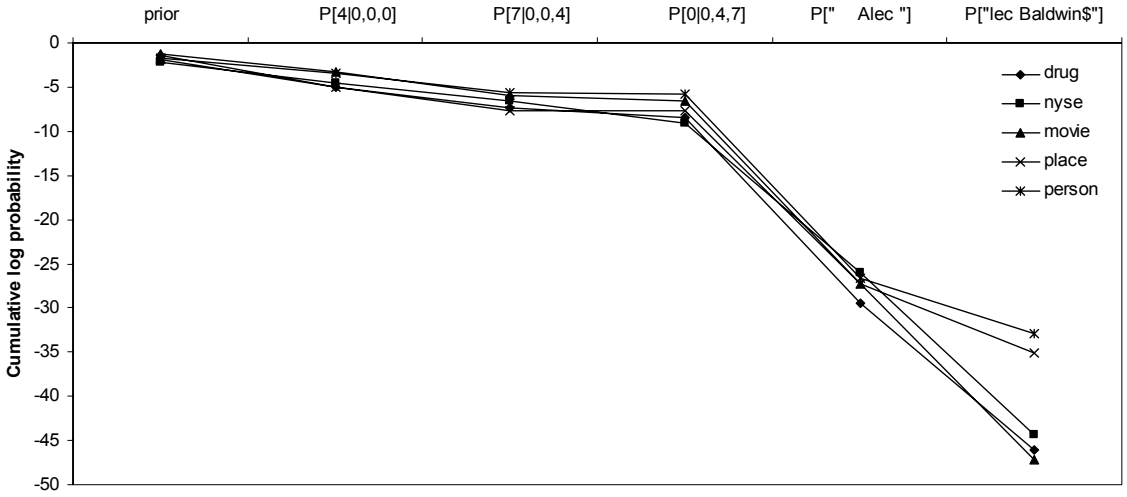

Figure 5. Cumulative classification scores for *Alec Baldwin.*

The stages of classification (x-axis) are prior probability, word-length sequences, and character-level sequences for each word. The running model score (cumulative log probability) for each category is shown along the y-axis. Higher values indicate stronger category resemblance.

Figure 6 shows a blow up of the last computation in Figure 5—the character-by-character scoring of the word *Baldwin*. Note that at the beginning of the word, the conditioning context includes the end of the previous word. This captures word-level collocations in addition to modeling word-internal regularities. Since *Baldwin* has been seen as both a person and place name but nothing else, the scores in those categories are much higher than in the other categories. Length-normalization has little effect here, but mixing in the word score at the end gives person a little additional bump, since the entire word was seen before in addition to having seen the substrings.

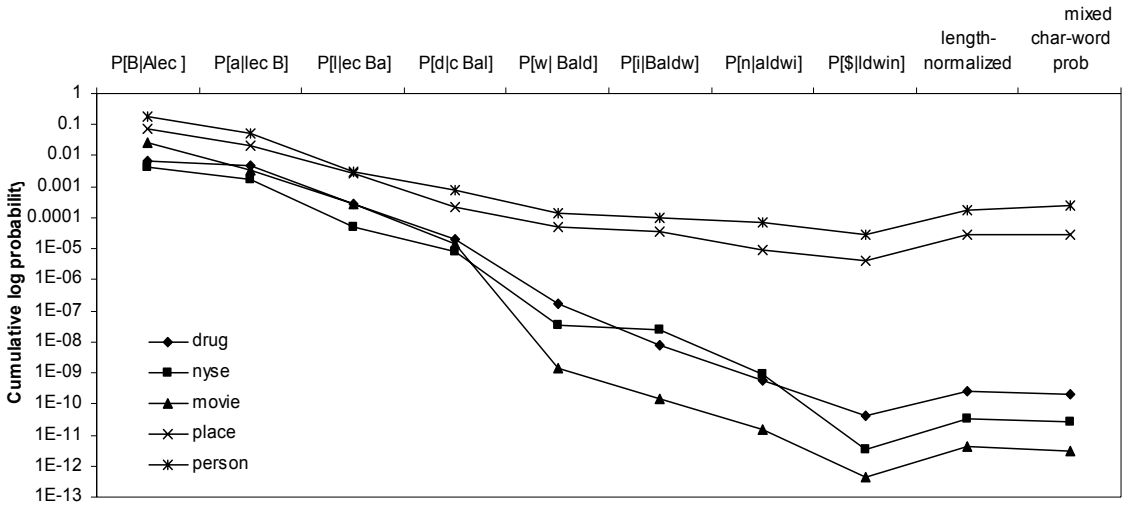

Figure 6. Cumulative classification scores for *Baldwin* (part of calculation above).

This is a blowup of the last column of Figure 5, showing the cumulative model score (log prob) for each category while a single word is processed one character at a time. The final two columns are normalizing to word-length (so short words and long words are weighted equally) and boosting the score for character sequences that happen to be a previously observed word.

Chapter 3:

Experiments in classifying proper noun phrases

This chapter utilizes the PNP model in Chapter 2 to perform a number of classification tasks. The primary results (and subsequent analysis in Chapter 4) are obtained on the five PNP categories described in Section 3.1. We subsequently present classification results on a wide variety of other PNPs in an attempt to showcase the versatility of our method and to gain a more comprehensive impression of its strengths and weaknesses.

3.1. Data sets used in experiments

We assembled five categories of PNPs for our primary set of experiments, each containing several thousand examples (see the appendix for complete information on counts and sources). The categories were pharmaceutical drugs (*drug*), companies listed on the New York Stock Exchange (*nyse*), movies and TV shows produced in 2000 (*movie*), cities and countries from around the world (*place*), and famous people's names (*person*). These collections were selected because they represented major sources of unknown PNPs of interest, and because of their diverse composition.

These data sets were intentionally left in the rather "noisy" state in which they were found, to breed robustness, and to accurately measure performance on "real world" examples. There is inconsistent use of capitalization, punctuation, and canonical formatting. Many of the PNPs within a given category come from different languages (e.g., foreign films). Some categories contain a number of frequently occurring words (e.g., *Inc.* and *Corporation* in *nyse*); others do not.[4]

It has been pointed out in the MUC competitions that PNPs often appear abbreviated in text, especially when mentioned earlier in full form. In such cases, not all of the information contained in these data sets would be available. Previous work in named entity extraction has addressed this problem by first trying to find full PNPs, then looking for their abbreviated versions (Mikheev et al. 1998). However, we note that the

[4] One thing we did manually correct was names that appear with their words in a non-standard order used for indexing, such as movie titles like *Ghost, The* and names like *Adams, John Quincy*. Each of these cases was restored to its "natural" word order.

current system can also recognize single-word unknown PNPs directly. Over 30% of the total PNPs used in these data sets are single-word PNPs, and in some categories, the number is much higher (e.g., 84% of place names are single words). Thus the ability of the classifier to handle both the presence and absence of common peripheral words in PNPs is being directly measured in our results.

3.2. Classifying drugs, companies, movies, places, and people

To assess the accuracy of our classifier, we ran three types of tests: *pairwise* tests of a single category against another single category, *1-rest* tests of a single category against the union of the other categories, and *n-way* tests, where all categories are against each other. The results of these tests are presented in Figure 7, sorted by classification accuracy and shown with standard deviations (computed from the ten separate train/test runs carried out for each result).

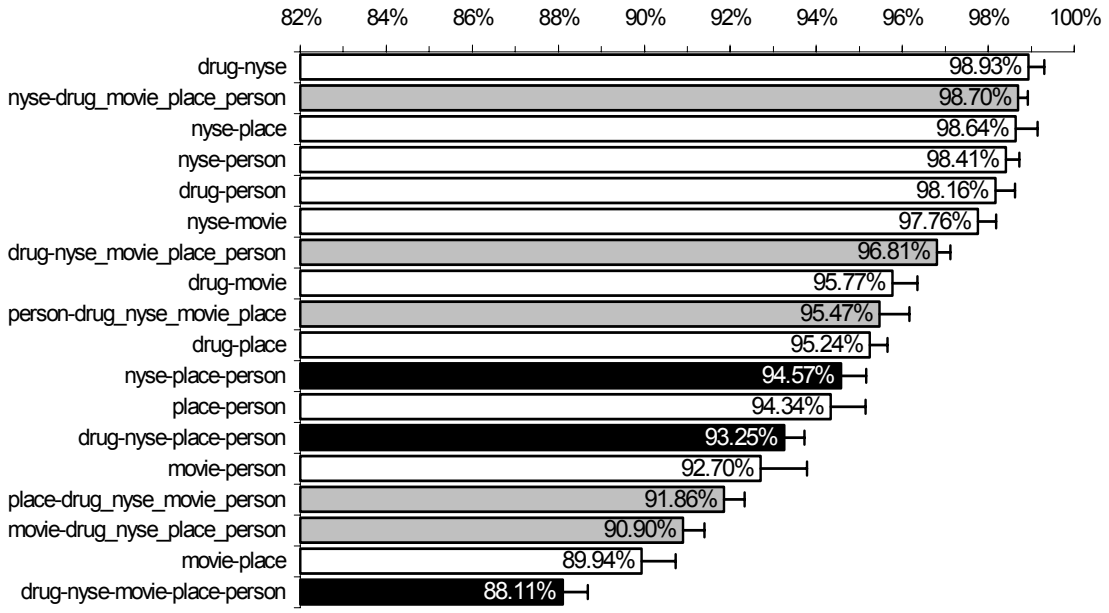

Figure 7. Classification accuracy on *pairwise* (white), *1-rest* (gray), and *n-way* (black) tests. Results are sorted by classification accuracy. The order reflects both the difficulty of the classification task (number of categories involved) and the difficulty of the input (which classes are being compared).

As expected, the *n-way* test was the most difficult. The ranking of results also reflects the inherent difficulty of the different categories. Overall, company names were most easily recognized, followed by drug names, person names, and place names, with movie titles proving the most difficult. These results are discussed further detail in Chapter 4.

3.3. Generalizing to new categories

The fact that our classification technique is domain independent means that we should be able to quickly retarget it for categories that have not previously been studied. We illustrate the flexibility of our approach here, by applying the classifier to a novel domain, without making any changes to the classifier design.

Inspired by the game "Cheese or Disease?", featured on the MTV show *Idiot Savants* (see Holman 1997), as our novel categories we chose discriminating names of cheeses and diseases. The basic idea is that there are many odd-sounding cheese and disease names, some of which are quite difficult to tell apart (e.g., *Yaws*, *Cerney*, *Orla*, *Hamartoma*)[5]. This seemed like an ideal test for the PNP classifier, since many of the names are single unknown words.

Finding existing lists of cheeses and diseases on the web proved easy (see Appendix for details), and since the classifier is domain independent, we were able to start training immediately. With ten minutes of work, we had a classifier that achieved 93.5% classification accuracy. Although recognizing references to cheese may not be high on the Defense Department's priority list, we feel that the ability to quickly reach high levels of proficiency in novel domains is a key benefit of our approach.

In the remaining sections of this chapter, we present further experiments classifying different categories of PNPs. We conclude with a preliminary evaluation of human accuracy at PNP classification as a benchmark for success. The goals in presenting these additional experiments are to investigate the adaptability of the classification method and to identity any systematic causes of errors.

[5] *Cerney* and *Orla* are cheeses; *Yaws* and *Hamartoma* are diseases.

3.4. Distinguishing biological names (genes, proteins, etc.)

Identifying and classifying PNPs in biomedical texts has become an increasingly important research topic in bioinformatics (Kazama et al. 2002). Names for proteins, genes, viruses, cells, and so on continue to appear at a frenetic pace, and there are no widely followed standards for terminology. Names are often quite opaque. e.g., *CD28* (protein) or *peri-kappa B site* (DNA), and they are also ambiguous, e.g., *E1A* can refer both to a gene and its protein product. It is therefore widely believed that biological PNPs are harder to identity and classify than traditional PNPs like the ones studied so far (Nobata et al. 2000). A major cause of this difficulty is that standard word-internal features like capitalization or presence of numbers are far less discriminating in biomedical text (where such "odd-looking" names are common to many categories) than in standard news text. Since switching from gross unknown word features to character-level features was successful in standard PNP classification tasks, there is reason to believe it might also provide benefit in the biomedical domain.

To investigate the ability of the PNP classifier to distinguish between different types biological names, we used the GENIA corpus, version 3.01 (Ohta et al. 2002). The GENIA corpus is one of the largest publicly available collections of annotated MEDLINE abstracts (2000 documents containing roughly 50,000 annotated name tokens), and it distinguishes 24 different biomedical name types, including *peptide*, *protein*, *nucleotide*, *DNA*, *RNA*, *carbohydrate*, *lipid*, *virus*, *tissue*, and so on. Much of the earlier work on biomedical named entity recognition (NER) was done on much smaller data sets (around 100 documents, e.g., Nobata et al. 1999, Collier et al. 2000) and this is seen as a contributing factor to the lack of high performance systems to date.

We trained our PNP classifier on all the annotated GENIA names using the experimental method described in Chapter 2. Average accuracy on the 24-way classification task was around 74%. A naïve majority-class baseline (which would call everything a protein) achieves 25%.

There are no directly comparable numbers in the literature for 24-way GENIA PNP classification, but we can get a rough comparison by considering some closely related work. Nobata et al. 1999 report PNP classification results on four GENIA categories (source, protein, DNA, and RNA) and their best system achieves 87.7-90.1%

accuracy. However, this is a much simpler task, because of the small number of classes.[6] Most results presented in the literature on biomedical NER combine the *identification* of names in free text (i.e. *segmentation*) and their subsequent *classification* (we consider the complete NER task in Chapter 5). When Nobata et al. add a segmentation algorithm (rather than classifying perfectly pre-segmented PNPs, their best results drop to 66.24%, a relative loss of 26% accuracy. Kazama et al. (2002) present results on all 24 GENIA categories, but only for the combined segmentation/classification task. Their best score is 50.23%.[7] If we assume that segmentation accounts for roughly 26% of the error (as it did for Nobata et al.), we can predict that they would obtain a 24-way classification accuracy of 63%. This is at best an informed guess, since the two systems use different segmentation methods. Nevertheless, it shows that 74% accuracy for our PNP classifier is reasonably state-of-the-art performance, and suggests that character-level features may be quite beneficial for distinguishing biomedical names.

3.5. Distinguishing bands and artists

There has been great consumer interest in recent years in digitizing one's entire CD collection and keeping all the music on a computer. The advantages are instant playback (no shuffling around for physical CDs) and complete flexibility in what songs are played in what order. However, as the quantity of digital music on computers continues to grow, it becomes increasingly important to organize and index it so people can quickly find what they are seeking.

Consider the task of alphabetically indexing all of the musical artists in your CD collection. It is a common practice to index band names by their first word (e.g., *Pink Floyd* goes under *P*), but to index artist names by their last name (e.g., *Eric Clapton* goes under *C*). This creates a challenge for an automatic indexing system: band names must be distinguished from artist names, since they are indexed differently. In some cases, this is relatively easy—for example, band names with *the* in their name (e.g., *The White Stripes*, *Huey Lewis and the News*, etc.) are easily distinguished. However in many cases

[6] While we did not have access to the *source* names, a 3-way classification between protein, DNA, and RNA using the PNP classifier achieves 87.1%, apparently a comparable result.

[7] Interestingly, their best result comes from using a form of substring features!

one simply has a two-word name and it is not obvious what to do with it. For example, *Crystal Method* is a band, but *Crystal Lewis* is an artist.

We investigated whether the PNP classifier could be used to distinguish band names from artist names. Luckily, the Google directory has an index of about 4500 bands (see Appendix for details) that have been correctly indexed (e.g., *Tori Amos* is listed as *Amos, Tori*). This makes it trivial to separate band and artist names and provides ample training data. For the experiments, artist names were reversed back to their original order (e.g., *Tori Amos*) and band names with *and* or *the* in their titles were set aside as too easy, leaving around 4000 ambiguous names that split roughly equally into band names and artist names.

We trained and tested the PNP classifier using the same experimental methods as above. Average classification accuracy was 90%. Most of the errors made were on names that are indeed quite ambiguous, and some a bit dubious. For example, the three "band names" that were misclassified as artist names with the highest confidence are *Luis Miguel*, *Dorian James*, and *Lynrd Skynrd*, all of which look like artist names (and we believe *Luis Miguel* was incorrectly labeled). If band names with *and* or *the* in their titles are included (as would be the case in a production system), accuracy increases to 95%.

An obvious idea for further improving accuracy is to add the *person* names from the previous set of experiments as additional training data. We added roughly 6000 extra "artist" names to each training fold and reran the experiments (ignoring the prior, which was now skewed). The performance gain was surprisingly negligible (1-2% on average). An error analysis suggests that most of the artist names that the system failed to recognize had uncommon names that were not found in the additional list of people's names. Thus the extra data did not tell the system much that it did not already know

In addition to the speed with which a reliable "smart indexing" system could be built with the PNP classifier, another benefit of this approach is that each classification decision comes with an attached probability that can be used as a confidence estimate. Depending on the confidence of the decision, different actions can be taken. For example, names that are highly ambiguous can be indexed twice, once normally and once reversed (e.g., index both *Luis Miguel* and *Miguel, Luis*). Assuming that the cost of indexing a name twice (where one instance is incorrect) is less than the cost of indexing it

once in the wrong place (where it can not be found), this is a reasonable strategy to pursue.

3.6. Distinguishing car models and computer models

Modern marketing has produced a plethora of synthetic product names—brand names made from combinations of made-up (or rare) words and numbers. Two product categories in which this can be seen clearly are car models (e.g., *Boxster*, *GS 400*, *MR2 Spyder*, *Passat*, *XK8*, etc.) and computer models (e.g., *Aptiva*, *Deskpro EN*, *Presario 7588*, *Z1*, *Vaio*, etc.). To car and computer aficionados, these names all sound distinct and meaningful (*XK8* is a Jaguar sports car, *Vaio* is a Sony laptop, etc.) but to an inexperienced observer, they appear qualitatively quite similar. For example, *Presario* (a Compaq desktop PC) has a rather car-like sound, similar perhaps to *Prius* or *Paseo*. Similarly, many brand names are just number and letter codes (especially luxury cars), e.g., *S500* (Mercedes), *750i* (BMW), or *V3100* (Toshiba desktop). Note however, that these codes are often internally consistent at the character level (e.g., there are many "S-class" Mercedes models, all with model numbers of the form *S###*). An added challenge is that the number of car and computer brand names, while certainly not insignificant, is considerably less than the other categories investigated (we collected roughly 700 car models and 150 computer models, compared to 1-10 thousand examples each of drugs, companies, locations, etc.). Thus it is an interesting empirical question whether our PNP classifier can distinguish between car and computer models given their apparent similarity and the lack of abundant training data.

Using the same train/test methodology above, our PNP classifier achieved an impressive accuracy of 98.1%, typically missing around 4 examples per 212-example test set, evenly split between classes. Most errors were due to misleading character sequences seen in the opposite class during training. For example, in one fold *OptiPlex GX150* (Dell desktop) was mistaken for a car model, because of the *Optima* and *GX 470* car models seen during training. Similarly, the computer model *Concerto III* was sometimes mistaken for a car because of car models like the *Jetta III*, *Golf III, and Bronco II*. Nevertheless, these cases were the clear exception, and the overall performance of the PNP classifier was on par with most Silicon Valley executives.

3.7. Comparison to human-level performance

In order to put the performance of our classifier in perspective, we tried to ascertain how good people are at this task. To measure this, we created an interactive version of the classifier in which examples from the pairwise (1-1) trials were presented one at a time and human subjects labeled the categories (we chose the 1-1 trials because they seemed the easiest and least confusing). The test consisted of six trials (one for each pair of drug, company, movie, and place names), each with 100 examples. The computer of course did over 1000 examples per trial, but humans have limited patience. We ran three subjects on each of the six trials, and recorded individual and average performance.

1-1 Trial	Avg. Human Accuracy	Computer Accuracy	Difference (C – H)
Drug-NYSE	97.67%	98.93%	+1.26%
Drug-Movie	93.00%	95.77%	+2.77%
Drug-Place	87.00%	95.24%	+8.24%
NYSE-Movie	97.00%	97.76%	+0.76%
NYSE-Place	93.33%	98.64%	+5.31%
Movie-Place	93.67%	89.94%	-3.73%
Average	***93.61%***	***96.05%***	***+2.44%***

Table 1. Human and computer performance at pairwise PNP classification.

Human results presented are the average of three subjects' performance on pairwise tests with 100 names per test. Computer results are taken from Figure 7.

While one should always be somewhat skeptical of an N=3 survey, there seems to be a clear trend in human performance, and in fact the results across subjects were very consistent. It is humbling to realize that in average performance, the computer is winning against Stanford students by 2.44%, which at this level of performance means that it is making roughly 40% fewer errors (though the set of human and computer errors are not necessarily the same). The difficulty of the different trials also seems to show through for both humans and the computer. The two notably unequal scores are *drug-place*, where the computer dominated and *movie-place*, where people won out decisively. Subjectively, *drug-place* was hard for people because there were lots of weird-looking and confusing one-word place and drug names. In contrast, *movie-place* was a trial where "higher-level semantic information" like what types of names tend to occur in movies vs. places seemed to help a lot. In conclusion, we feel it is fair to claim that our classifier performs at or near the human level of competence.

Chapter 4:

Analysis of experimental results

In this chapter, we account for the experimental results presented in Chapter 3, analyze the contribution of each model component to overall performance, and examine the various parameters learned during cross validation. This discussion is concerned primarily with the main five categories described in Section 3.1.

4.1. Sources of erroneous classification

Figure 8 shows the confusion matrix for the *five-way* classification task. The area of each circle is proportional to the number of examples in that cell. Movies, places, and people are most often confused for one another, and drugs are often misclassified as places (since they both contain many odd-looking one-word names). As an indication of the difficulty of dealing with movie titles, when the *n-way* tests are rerun as a *four-way* test without movie titles, average classification accuracy jumps from 88.1% to 93.2%. *Three-way* classification between companies, places, and people (similar to the ENAMEX classification task in MUC) is performed with an average accuracy of 94.6%.

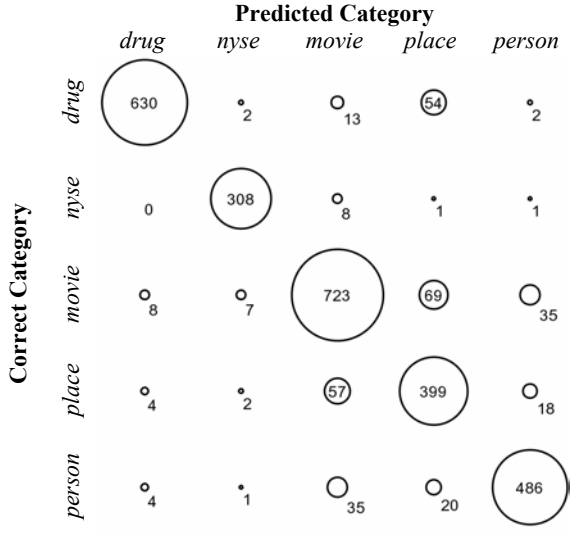

Figure 8. Confusion matrix for 5-way test.
The area of each circle is directly proportional to the number of counts.

The ease of identifying company names is largely attributable to the plethora of common words available, such as *International*, *Capital*, *Inc.*, and *Corporation*. The difficulty of place names and movie titles is partly due to the fact that they contain words from many different languages (and thus the learned estimates blur together what should really be separate distributions). Movie titles are also the most inherently ambiguous, since they are often named after people (e.g., *John Henry*) or places (e.g., *Nuremberg*), and often contain words normally associated with another category (e.g., *Prozac Nation* and *Love, Inc.*). Mikheev et al. (1998) report instances of similar ambiguity as a cause of error in their work.

A similar source of errors stems from words (and common intra-word letter sequences) that appear in one category and drive classification in other categories when there is insufficient information to the contrary. For example, in one run *Delaware* was erroneously classified as a company, because it was never seen as a place name, but it was seen in several company names (such as *GTE Delaware LP*). The same phenomenon was the major cause of errors in the car-computer experiment presented in Section 3.6. Cases like these appear to be an inherent limitation of our approach. However we are being unusually restrictive by forcing our test set to be completely disjoint with our training set. In a real application, common place names like *Delaware* would have been trained on and would be readily recognizable as place names.

The reader is challenged to classify five PNPs—*R & C*, *Randall & Hopkirk*, *Steeple Aston*, *Nandanar*, and *Gerdau*— in order to gain an appreciation for the potential difficulty of this task (see Appendix for solutions):

4.2. Contribution of model features

To assess the relative contribution of the three major components of our model (length *n*-gram, character *n*-gram, and common words), we present accuracy results using each model component in isolation and each possible pair of components (Figure 9).[8] We use the *four-way* test *drug-nyse-place-person* as a representative indicator of performance.

[8] Note that the full model in Figure 3 is identical to the *four-way* test in Figure 1. The slight difference in performance is merely due to data set differences.

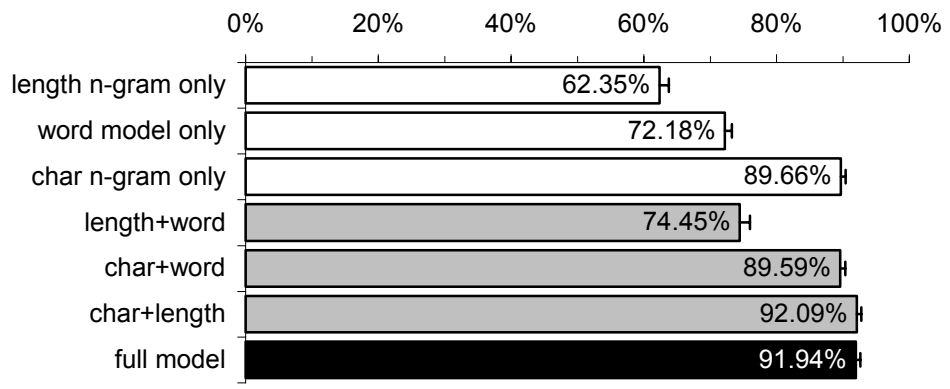

Figure 9. Classification accuracy for individual model components and combinations (*four-way* test).

Each feature gives a classification accuracy significantly above a most-frequent-class baseline (34%), with the character *n*-gram being by far the most powerful single feature. Combining features reveals that the character and length *n*-grams provide complementary information, but adding the word model to the character *n*-gram does not improve performance. The common word model by itself is quite effective, but it is largely subsumed by our high order character *n*-gram model, because common short words are memorized as single *n*-gram entries, and common long words contain many common *n*-grams. The word model could be eliminated without hurting performance.

The word model could also be regarded as a reasonable baseline, since it is basically equivalent to the performance expected from a (multinomial) Naïve Bayes word model, a model that is often used as a baseline in text classification tasks. As one further indicative baseline, we ran a publicly available variable *n*-gram language identifier on our data (Beeferman 1996), which achieves an accuracy of 76.54%. This is not a fair comparison: Beeferman explicitly notes that his system is unlikely to be reliable on very short inputs of the sort present in our data, but this nevertheless again shows that our system is sufficiently well tuned to the task at hand to achieve performance well above obvious baseline levels.

4.3. Impact of other model parameters

In addition to the three major model components described in Section 4.2, performance is affected by the length of the *n*-gram models used, the use of a word length

normalization constant for the character *n*-gram, and the amount of available training data.

4.3.1. Increasing *n*-gram length

The only important model parameters not set on held-out data are the sizes of the length and character *n*-gram models. In principle, the use of deleted interpolation with weights set on held-out data means that very large *n*-gram models could be used, and once data sparseness was a larger factor than predictive accuracy, the higher-order *n*-gram factors would be down-weighted. However in practice training and testing is exponentially slow in the length of the *n*-gram, and the largest useful *n*-gram size, once determined empirically, is relatively stable.

Table 2 shows classification accuracy of the character *n*-gram model alone for increasing values of n. Accuracy increases and plateaus, at which point increasing n further has no effect. The same analysis holds for increasing n for the word-length *n*-gram, also shown in Table 2.

n	1	2	3	4	5	6	7
char	60.0	81.4	87.7	89.2	89.5	89.7	89.8
length	46.4	60.1	62.2	62.4	62.4	-	-

Table 2. Classification accuracy of char and word-length *n*-gram models alone. Results are presented on the *four-way task.* Accuracy for the 6- and 7-gram length model were not obtained because performance had already reached a plateau.

4.3.2. Word-length normalization

As mentioned above, modeling words with a character *n*-gram model means treating characters as the unit of evidence, so that long words have more of an impact on classification than short words. However, in many instances, it seems intuitively clear that words are a better unit of evidence (indeed, many telling common words like *Inc.* or *Lake* are very short). To compensate for this effect, we introduce a parameter to normalize the probability assigned to each word in a PNP by taking the (*k*/*length*)'th root, where *length* is the number of characters in the word, and *k* is a global constant set with a line search on held-out data.

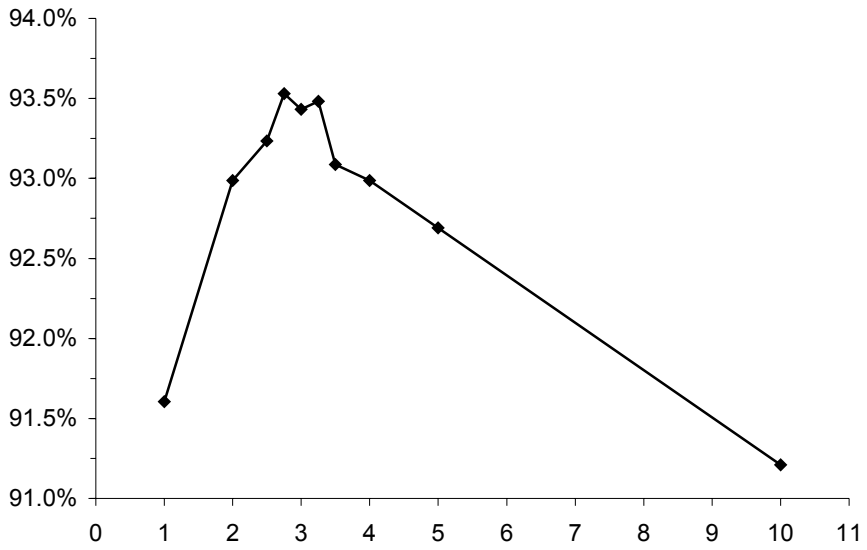

Figure 10. Classification accuracy vs. word length normalization constant (4-way test).

When characters are the unit of evidence, longer words will naturally have a greater effect on the overall classification decision. The role of length-normalization is to mitigate this effect—words of length below the cutoff will have their scores boosted while those above the cutoff will have it diminished. The optimal tradeoff is an empirical property of the given classification task.

Figure 10 shows how varying the value of *k* affects performance on a typical run. The optimal value of *k* varies by data set, but is usually around 2 to 3. Probability judgments for words of length < *k* are magnified, while those for words of length > *k* are diminished. The result is that compelling short words can effectively compete with less compelling longer words, thus shifting the unit of evidence from the character to the word.

4.3.3. Training data size

Obtaining a large number of examples of each category to use as training data was not difficult. Nevertheless, it is still worth examining classifier performance as a function of the amount of training data provided. Figure 11 illustrates that, while performance continues to improve as more training examples are provided, the classifier only requires a small subset of the training data to approach its full potential. This means that, when faced with novel categories for which large collections of examples are not immediately available, acquiring proficiency should still be possible.

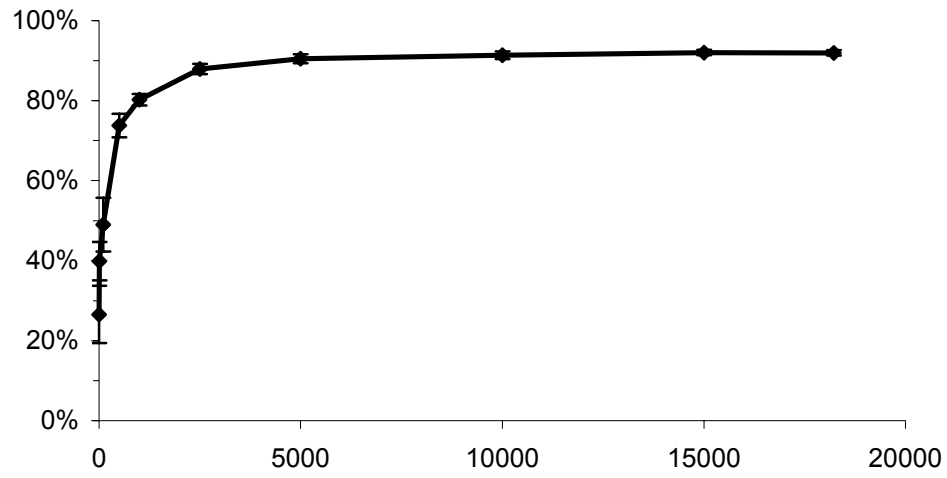

Figure 11. Classification accuracy vs. number of training examples (*four-way* test). Error bars at each point show standard deviation from 10 trials (hardly visible).

Figure 11 also indicates that increasing the amount of training data would not significantly boost performance. This hypothesis is supported by the observation that the majority of misclassified examples are either inherently ambiguous, or contain words that appeared in another category, but that are not strongly indicative of any one category (as mentioned in Section 4.1).

4.4. Generation of novel PNPs

Generative models are common for classification, but can also be used to perform generation. We can stochastically generate a collection of novel PNPs as an alternative means for getting a sense of the quality of the learned models. First we generate freely from the word-length *n*-gram model until a 0-length word (special stop token) is generated. This gives us a "template" for each word in the PNP (e.g., `5-3-0`). Next, each word is filled in by generating from the character *n*-gram model. Since it would be unnatural to force a space after a predefined number of characters (e.g., to force the first word to be five characters long), we employ rejection sampling: words are generated and discarded until the first one is naturally generated with the desired length. The generation of subsequent words starts with the ending context of the previous word, as is done for

classification. A favorable selection of generated examples for the five main categories is presented in Table 3.

Drug:	*Esidrine Plus Base with Moisturalent • Ambenylin • Carbosil DM 49*
NYSE:	*Downe Financial Grp PR • Intermedia Inc. • Host Manage U.S.B. Householding Ltd.*
Movie:	*Dragons: The Ever Harlane • Alien in Oz • El Tombre*
Place:	*Archfield • Lee-Newcastleridge • Qatad*
Person:	*Benedict W. Suthberg • Hugh Grob II • Elias Lindbert Atkinson*

Table 3. Sample of artificially generated proper noun phrases in several categories.

Not all generated examples are this coherent, but we are encouraged by the surprisingly natural look of a large fraction of what is generated. Sometimes entire training examples are generated, but usually the result is a mix of existing and novel words mixed together. In general, the closer that stochastically generated examples are to real examples, the closer the model is capturing the "essence" of the proper name distribution.

Chapter 5:
Combing segmentation and classification

While proper noun phrase classification on its own can be a valuable application, most commonly it takes place alongside finding and segmenting the PNPs in continuous text (i.e. highlighting all the names in a document and annotating them with their semantic category). This chapter describes several extensions to the basic model that allow for integrated segmentation and classification. In some systems, segmentation and classification are performed as separate, isolated steps (Florian 2002, Cucerzan & Yarowsky 2002). However, we prefer to view the task as one of sequence classification, in which each word is labeled as either background text or one of several PNP categories.

The work presented here is largely based on work by Klein, Smarr, Nguyen, and Manning (2003). Our experiments were conducted using the CoNLL 2003 shared task dataset (Tjong Kim Sang & De Meulder 2003), which requires identifying people's names (PER), organization names (ORG), place names (LOC), and miscellaneous other names (MISC) in news text. In this domain, PNPs are referred to as "named entities" and the task of segmenting and classifying PNPs is referred to as "named entity recognition" (NER). Data sets were available for both English and German text. In both languages, the data sets were divided into a large training set, a small development set for testing and tweaking the model, and a small test set for reporting final scores. Words came pre-tokenized, and were annotated with their part of speech and chunk tag (though we ignored the latter).

The two key questions this chapter addresses are: (1) whether character-level PNP classification methods effective when the PNPs don't come pre-segmented; and (2) whether the benefits of using character-level features for PNP classification are complementary to the value from external contextual cues one has access to when classifying PNPs embedded in text.

5.1. Challenges for named entity recognition

Finding a known named entity (e.g., *Stanford University*) in text is fairly simple because you can just look for an exact phrase match (although names are often

abbreviated, e.g., *Stanford* may appear alone). However, highlighting unknown named entities is considerably more difficult, because they must be recognized as names, and their boundaries must be accurately determined. In English, capitalization provides strong evidence for the existence of a named entity (e.g., "*...while staying in Great Britain for the summer...*") although this can be misleading at the beginning of sentences or in titles and headlines (e.g., "*Great British restaurants are hard to find"* or "*Will Great Britain Welcome the Euro?*"). In German, all nouns are capitalized, making the task much more difficult (since only proper nouns are of interest).

Most research has addressed the unknown word problem in NER just as described in Section 1.3—with a collection of syntactic context features and gross word-internal features such as suffixes, capitalization, and punctuation 1997, Wacholder et al. 1997, Bikel et al. 1997). Given the relative ease of recognizing known names and the relative difficulty of recognizing unknown names, we believe that focusing on unknown entities is the key challenge for robust NER. Thus our approach in this chapter is a natural extension of the work presented in the previous chapters, both in terms of the task definition and the motivation for the particular solution we adopt.

As was in the case in the PNP classifier, for NER we adopt character sequences as a primary representation. We present two models in which the basic units are characters and character *n*-grams, instead of words and word phrases. Earlier papers have taken a character-level approach to named entity recognition, notably Cucerzan and Yarowsky (1999), which used prefix and suffix tries, through to our knowledge, incorporating all character *n*-grams is new. In Section 5.2, we discuss a character-level hidden Markov model (HMM) that uses the PNP classifier to model emissions, while in Section 5.3, we discuss a sequence-free maximum-entropy (maxent) classifier that uses *n*-gram substring features. Finally, in Section 5.4, we add additional features to the maxent model, and chain these models into a conditional Markov model (CMM), as used for tagging (Ratnaparkhi 1996) or earlier NER word (Borthwick 1999). We use the English development set to compare the performance of different models and feature sets, and we report test set performance on English and German at the end using our best model.

5.2. A character-level HMM

Figure 12 shows a graphical representation of our character-level HMM. Characters are emitted one at a time, and there is one state per character. Each state's identity depends only on the previous state. Each character's identity depends on both the current state and on the previous $n - 1$ characters. In addition to this HMM view, it may also be convenient to think of the local emission models as type-conditional n-gram models. Indeed, the emissions are modeled using the character n-gram component of the PNP classifier described in Chapter 2. The primary addition is the state-transition chaining, which allows the model to do segmentation as well as classification.

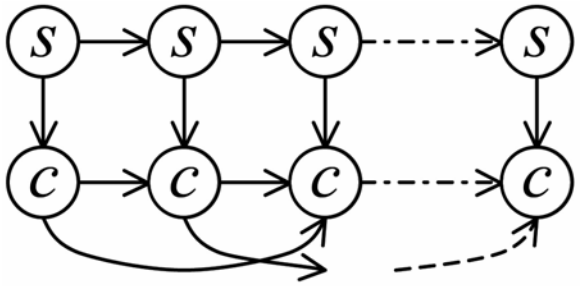

Figure 12. A character-level HMM.
The *c* nodes are character observations and the *s* nodes are entity types.

When using character-level models for word-evaluated tasks, one would not want multiple characters inside a single word to receive different labels. This can be avoided in two ways: by explicitly locking state transitions inside words, or by careful choice of transition topology. In our current implementation, we do the latter. Each state is a pair (t, k), where t is an entity type (such as *PERSON*, and including an *other* type) and k indicates the length of time the system has been in state t. Therefore, a state like (*PERSON*, 2) indicates the second letter inside a person phrase. The final letter of a phrase is a following space (we insert one if there is none) and the state is a special final state like (*PERSON*, F). Additionally, once k reaches our n-gram history order, it stays there. We then use empirical, unsmoothed estimates for state-state transitions. This annotation and estimation enforces consistent labelings in practice. For example, (*PERSON*, 2) can only transition to the next state (*PERSON*, 3) or the final state

(*PERSON*, F). Final states can only transition to beginning states, like (*other*, 1). For emissions, we must estimate a quantity of the form $P(c_0|c_{-(n-1)}, \ldots, c_{-1}, s)$, for example, P(s|Thoma, PERSON, 6).[9] This is accomplished using the PNP classifier in a *single-class* mode in which each state only emits one kind of PNP (i.e. it just represents a character-level distribution of a set of strings).

Given this model, we can do Viterbi decoding in the standard way (Viterbi 1967). To be clear on what this model does and does not capture, we now consider a few examples (_ indicates a space). First, we might be asked for P(e|to_Denv, *LOC*, 5). In this case, we know both that we are in the middle of a location that begins with Denv and also that the preceding context was to. In essence, encoding *k* into the state lets us distinguish the beginnings of phrases, which lets us model trends like named entities (all the classes besides *other*) generally starting with capital letters in English. Second, we may be asked for quantities like P(_|Italy, *LOC*, F), which allows us to model the ends of phrases. Here we have a slight complexity: by the notation, one would expect such emissions to have probability 1, since nothing else can be emitted from a final state. In practice, we have a special stop symbol in our *n*-gram counts, and the probability of emitting a space from a final state is the probability of the *n*-gram having chosen the stop character.[10]

Using this model, we tested two variants, one in which preceding context was discarded (for example, P(e|to_Denv, *LOC*, 5) was turned into P(e|XX_Denv, *LOC*, 5), and another where context was used as outlined above. For comparison, we also built a first-order word-level HMM. The results, shown in Table 4, give F_1 accuracies both per-category and overall. The word-level model and the (context disabled) character-level model are intended as a rough minimal pair, in that the only information crossing phrase boundaries was the entity type, isolating the effects of character- vs. word-level modeling (a more precise minimal pair is examined in 5.3). Switching to the character model

[9] We index characters, and other vector elements by relative location subscripts: c_0 is the current character, c_1 is the following character, and c_{-1} is the previous character.

[10] This can be cleaned up conceptually by considering the entire process to have been a hierarchical HMM (Fine et al. 1998), where the *n*-gram model generates the entire phrase, followed by a tier pop up to the phrase transition tier.

raised the overall score greatly, from 74.5% to 82.2%. Use of context provided additional benefit, but substantially less, bringing the total to 83.2%.

Description	ALL	LOC	MISC	ORG	PER
Official baseline	71.2	80.5	83.5	66.4	55.2
Word-level HMM	74.5	79.5	69.7	67.5	77.6
Char-level, no context	82.2	76.1	82.2	73.4	84.6
Char-level, context	**83.2**	**86.9**	**83.0**	**75.1**	**85.6**

Table 4. HMM F_1 performance on English development set.

5.3. A character-feature based classifier

Given the amount of improvement from using a model backed by character *n*-grams instead of word *n*-grams, the immediate question is whether this benefit is complementary to the benefit from features which have traditionally been of use in word level systems, such as syntactic context features and topic features.

To test this, we constructed a maxent classifier that locally classifies single words, without modeling the entity type sequences *s*.[11] These local classifiers map a feature representation of each word position to entity types, such as *PERSON*. We present a successive hill-climb of results over added feature sets for the English development set data in Table 5. First, we tried only the local word as a feature; the result was that each word was assigned its most common class in the training data. The overall F-score was 52.29%, well below the official CoNLL baseline of 71.18%, which simply matches all full and unambiguous named entities found in the training data.[12]

We next added *n*-gram features; specifically, we framed each word with special start and end symbols, and then added every contiguous substring to the feature list. Note that this subsumes the entire-word features. Using the substring features alone scored 73.10%, already breaking the phrase-based CoNLL baseline, though lower than the no-

[11] The classifier was trained using conjugate gradient descent, used equal-scale gaussian priors for smoothing, and learned models of over 800K features in approximately 2 hours.

[12] Note that the word model actually outperforms the baseline model for people's names, because the latter is unable to synthesize new first-name/last-name pairs, while the former can do so freely.

context HMM, which better models the context inside phrases. Adding a current tag feature gave a score of 74.17%.

At this point, the bulk of outstanding errors were plausibly attributable to insufficient context information. Adding just the previous and next words and tags as (atomic) features raised performance to 82.39%. More complex, joint context features that paired the current word and tag with the previous and next words and tags raised the score further to 83.09%, nearly to the level of the HMM, without actually having any model of previous classification decisions.

Description	Added Features	ALL	LOC	MISC	ORG	PER
Words	w_0	52.29	41.03	70.18	60.43	60.14
Official Baseline	-	71.18	80.52	83.52	66.43	55.20
N-Grams	$n(w_0)$	73.10	80.95	71.67	59.06	77.23
Tags	T_0	74.17	81.27	74.46	59.61	78.73
Simple Context	w_{-1}, w_0, t_{-1}, t_1	82.39	87.77	82.91	70.62	85.77
More Context	$\langle w_{-1}, w_0\rangle$, $\langle w_0, w_1\rangle$, $\langle t_{-1}, t_0\rangle$, $\langle t_0, w_1\rangle$	83.09	89.13	83.51	71.31	85.89
Simple Sequence	s_{-1}, $\langle s_{-1}, t_{-1}, t_0\rangle$	85.44	90.09	80.95	76.40	89.66
More Sequence	$\langle s_{-2}, s_{-1}\rangle$, $\langle s_{-2}, s_{-1}, t_{-1}, t_0\rangle$	87.21	90.76	81.01	81.71	90.8
Final	**misc. extra features**	**92.27**	**94.39**	**87.10**	**88.44**	**95.41**

Table 5. CMM performance with incrementally added features on English development set.

5.4. A character-based CMM

In order to include state sequence features, which allow the classifications at various positions to interact, we have to abandon classifying each position independently. Sequence-sensitive features can be included by chaining our local classifiers together and performing joint inference, i.e., by building a conditional Markov model (CMM), also known as a maximum entropy Markov model (McCallum et al. 2000).

Previous classification decisions are clearly relevant: for example the sequence `Grace Road` is a single location, not a person's name adjacent to a location (which is the erroneous output of the model in Section 5.3). Adding features representing the previous classification decision (s_{-1}) raised the score 2.35% to 85.44%. We found knowing that the previous word was an *other* wasn't particularly useful without also knowing its part of speech (e.g., a preceding preposition might indicate a location). Joint tag-sequence features, along with longer distance sequence and tag-sequence features, gave 87.21%.

The remaining improvements involved a number of other features which directly targeted observed error types. These features included letter type pattern features (for example `20-month` would become `d-x` for digit-lowercase and `Italy` would become `Xx` for mixed case). This improved performance substantially, for example allowing the system to detect ALL CAPS regions. Other features included second-previous and second-next words (when the previous or next words were very short) and a marker for capitalized words whose lowercase forms had also been seen. The final system also contained some simple error-driven post-processing. In particular, repeated sub-elements (usually last names) of multi-word person names were given type *PERSON*. In total, this final system had an F-score of 92.31% on the English development set. Table 6 gives a more detailed breakdown of this score, and also gives the results of this system on the English test set, and both German data sets.

English dev.	Precision	Recall	$F_{\beta=1}$
LOC	94.44	94.34	94.39
MISC	90.62	83.84	87.10
ORG	87.63	89.26	88.44
PER	93.86	97.01	95.41
Overall	92.15	92.39	92.27

English test	Precision	Recall	$F_{\beta=1}$
LOC	90.04	89.93	89.98
MISC	83.49	77.07	78.85
ORG	82.49	78.57	80.48
PER	86.66	95.18	90.72
Overall	86.12	86.49	86.31

German dev.	Precision	Recall	$F_{\beta=1}$
LOC	75.53	66.13	70.52
MISC	78.71	47.23	59.03
ORG	77.57	53.51	63.33
PER	72.36	71.02	71.69
Overall	75.36	60.36	67.03

German test	Precision	Recall	$F_{\beta=1}$
LOC	78.01	69.57	73.54
MISC	75.90	47.01	58.06
ORG	73.26	51.75	60.65
PER	87.68	79.83	83.57
Overall	80.38	65.04	71.90

Table 6. Final results obtained for the development and test sets for each language.

5.5. Discussion

These results demonstrate that character-level features are indeed a useful source of information for NER. When using an HMM, switching from a word-level model to a character-level model reduces errors by 30%. Furthermore, this benefit appears to be complementary to that obtained through contextual evidence. Even in our final context-rich CMM, using character substrings instead of entire words still reduces errors by 25%.

Character-level models also appear to extend well to the task of segmentation. Using the PNP classifier in a character-level HMM means modeling background text at

the character-level as well as PNPs themselves. Given the heterogeneity and length of arbitrary background news text, it is perhaps surprising that this works as well as it does. However, as long as the PNPs come from a relatively distinct distribution, they can accurately identify their own words, leaving the background model to collect the rest, even if the latter does not have a clearly defined distribution of its own.

Two notable features of the results presented in table 6 are that English test performance is considerably lower than English development performance, and German performance in general is much lower than English performance. While a small drop from development to test is to be expected, in this case a large source of error was inconsistent labeling across the English data sets, particularly in inherently ambiguous places such as sports teams named for their home town. For example, "Boston" can refer to the city of Boston (LOC) or a sports team from Boston (ORG), and the annotation didn't consistently sort these cases out. The drop from English to German is mainly attributable to the lack of capitalization cues. The English results have roughly equal precision and recall, while in German, recall suffers more than precision. The reason is that, without capitalization in German, the system is dealing with weaker evidence in general, and thus only the most compelling examples will be tagged. As a result, precision (accuracy of tagged names) will remain reasonable, but recall (coverage of all tagged names) will suffer because weakly suggestive names are left as background text.

Chapter 6:
How and why proper names can be distinguished

The preceding chapters demonstrated that the composition of many proper names is far from arbitrary with respect to their meanings. This apparently contradicts the common assumption made throughout linguistics that the relationship is a historical artifact. It is thus worth speculating on the forces that shape the naming process. A better understanding of the process by which names are generated can serve both as an explanation for the success of the methods presented so far, and as an inspiration for improving them further. In this chapter, we consider relevant linguistic findings in sound symbolism, the relationship between PNP classification and language identification, and the modern business of creating new brand names.

6.1. Relation to sound symbolism

Linguists generally assume that, for the most part, the relationship between the *sounds* of words (or equivalently their written form) and their *meaning* is arbitrary (Saussere 1916/1974, Holdcroft 1991). In fact, this postulate is the foundation on which modern comparative linguistics is founded—language families are identified by finding similar sound-meaning pairs and assuming this must be the result of common linguistic origin (or sometimes borrowing, Ruhlen 1996). If the mapping between sound and meaning were completely arbitrary, then PNP classification would not be possible—there would be no statistical correlations that could be used to improve upon chance guessing. Given that, empirically, this is not the case, it is clear that, at least for proper nouns, there is some degree of *sound-symbolism* present. Sound symbolism is commonly thought of as concerned with rare cases such as onomatopoeia, but in fact it is a more general area of research on any relationship between the sound of an utterance and its meaning, and there is now evidence from a variety of sources that "sound symbolism plays a considerably larger role in language than scholarship has hitherto recognized (Hinton et al. 1994).

Research in sound symbolism has shown that people (implicitly) pick out relationships between phoneme classes and semantic classes that extend across many languages. For example, Hinton et al. (1994) report that experiments performed by

Berlin and LaPolla suggest that English speakers can correctly guess semantic components of Chinese and Jivaro words based on their phonemic composition (p. 10). Additional sound-symbolic tendencies exist within individual languages. For example, Sereno (1994) shows that, in English, nouns are more likely to have back vowels while verbs are more likely to have front vowels, and this difference is reflected in processing times for lexical decision tasks. Subjects were significantly faster at deciding whether a word was a noun or a verb when the vowel pattern was consistent (back vowels for nouns, front vowels for verbs) than in the reverse condition. This effect held for both high frequency and low-frequency words, even though the vowel-noun/verb relationship holds mainly for high frequency words.

Rhodes (1994) proposes that English has a rudimentary classifier system, which can be seen by examining sets of semantically related words with similar phonemic prefixes (reproduced from p. 276):

st-	[1 dimensional]	(*stick*, *staff*, *stem*, etc.)
str-	[1 dimensional, flexible]	(*string*, *strand*, *strip*, etc.)
fl-	[2 dimensional]	(*flap*, *flat*, *floor*, etc.)
š/sk-	[2 dimensional, flexible]	(*sheet*, *scarf*, *skin*, etc.)
n-	[3 dimensional]	(*knob*, *knot*, *node*, *nut*, etc.)
sp-	[cylindrical]	(*spool*, *spine*, *spike*, etc.)
dr-/*tr-*	[liquid]	(*drink*, *drain*, *tickle*, *trough*, etc.)
et al.		

While such systematic correspondences are striking, they are certainly not omnipresent. Leben (2003) points out for example that although the short /*i*/ vowel is evocative of slenderness (e.g., *thin* and *slim*) it is also found in words like *thick* and *big*. More generally, while universal sound-symbolic relationships do exist for many phonemes, these relationships are systematically violated and imported into infelicitous environments. Leben draws an analogy to body language—which an integral part of everyday communication and which can greatly enhance the ease of comprehension and richness of meaning of an utterance when used properly. However, it is not essential—books (and telephone conversations) make no use of body language and they are still adequately communicative. Similarly, sound-symbolism can aid interpretation and evoke additional associations, but many or most words make do without it.

In summary, the ability of our PNP classifier to identify the semantic class of unseen words without the aid of surrounding context is less surprising and magical when considered in the larger context of sound-symbolic processes in language. In the range of presence of sub-word sound-symbolic units, proper names may be among the most susceptible (especially artificially constructed names, discussed in Section 6.3), and our model may well be picking up on these implicit sound-meaning pairings in order to increase classification accuracy.

6.2. PNP classification as language identification

Language identification is the task of classifying a passage of text with the primary language in which it is written. It is conceptually similar to the task on PNP classification in that two strings of text, written with the same character set, nevertheless have a distinctive *look* to their character sequences that can be modeled and exploited for classification. In fact, PNP classification can be thought of as a type of language identification—distinguishing the "language" of company names, the "language" of drug names, and so on.

Perhaps not surprisingly, several successful language modeling techniques make heavy use of character-level statistics (Dunning 1994, Cavnar & Trenkle 1994). That is, rather than trying to identify a language by looking for common words (like "the" or "la"), they simply look for common (often sub-word) character sequences (like "-able" or "-esque"), which are more numerous and in fact often more distinct. Recent work (Peng et al. 2003) have generalized these character-level language models to perform a wide variety of text classification tasks such as authorship attribution (who wrote which passages), genre identification, and topic detection. They achieve impressive results on a wide variety of languages, sharing our experience in Chapter 5 that character-level models can be easily applied to many languages. One difference between our approach and most other work in language identification is that the latter generally depends on having considerably more text available for input than a single PNP. As we point out in Section 4.2, this helps explain why our PNP classifier significantly outperforms the character-level language identification system of Beeferman (1996).

6.3. The business of creating names

Our method for classifying proper names is to create a statistical model of their generation. A natural question, therefore, involves the extent to which our model is similar to the way in which new names are actually created. For names of people or places, most of which were created ages ago, this can be a difficult question to answer. But there is at least one place to look to for inspiration—companies whose primary business is coming up with new names for companies and products. In fact, there are a number of professional brand-name creators (Landor, Lexicon Branding, Cintara, medibrand, etc.), and their business is booming.[13] As Kahn (2001) puts it, "What's in a name…about 40 grand, give or take". Krauskfopf's (2002) gives even higher figures: "It takes pharmaceutical companies years and costs them about $2.25 million to name each medicine." The trick is finding a name that is distinctive enough to be legally protected, but that has a "product of several powerful sounds". *Prozac* is touted as one of the best invented drug names, and is one of the world's best-known and best-selling drugs ever. The Intel chip *Pentium* is considered the strongest technology brand of the 1990s, an asset worth millions (Leben 2003).

So how do naming consultants come up with new names? Part of the effort is free-form brainstorming, word- and morpheme-level semantic associations, and so on. But a large component in many cases is the sound-symbolically driven composition of meaningful character sequences. For example,

> Lexicon has completed extensive research into how sound symbolism affects the way brand names are perceived. If a product would be perceived as faster, bigger, or even more reliable depending on how it sounds, it follows that there would be an entirely new set of tools to add to the creative process. The results prove that there is.[14]

Exploiting people's tacit knowledge of sound symbolism has become a cornerstone of modern brand-name creation. For example, the popular handheld wireless email device *BlackBerry* gets its name because people "associate the *b* sound with reliability…while the short *e* evokes speed" (Begley 2002; the original proposed name was "Strawberry" because the little keyboard buttons look like seeds). Begley goes on to describe how "as

[13] See www.landor.com, www.lexiconbranding.com, www.cintara.com, and www.medibrand.com
[14] See http://www.lexiconbranding.com/process2aSound.html (visited: 5/4/2003)

winning hybrids of real words become scarcer…some naming consultants are advising brand managers to tap different synapses in their customers' brains: those linking the raw sounds of vowels and consonants to specific meanings and even emotions". At first this all may sound like a slick marketing pitch from the branding consultants, but several studies, including one by Yorkston and Menon (2003) have found that "consumers use information they gather from phonemes in brand names to infer product attributes and to evaluate brands" (p. 3). They conclude furthermore that "the manner in which phonetic effects of brand names manifest is automatic in as much as it is uncontrollable, outside awareness, and effortless." Similar findings are presented by Klink (2001). However, not everyone is a believer in the benefits of sound-symbolic name creation. As a blog (a self-published web journal) connected to the branding firm "A Hundred Monkeys" laments, "in the dark ages before linguistics got the upper hand, cars had names like *Corvette*, *Camaro*, *Mustang* and *Stingray*. Now that the professionals are in control, we have names like *Alero*, *Prius*, *Bravada*, and *Escalade*."[15]

While companies like Lexicon have conducted extensive research on the meanings associated with each phoneme, most of the knowledge that goes into actual brand name creation is tacit in the heads of naming consultants, not explicit in a generative computer model (Leben 2003). There are computer programs that will take as input a set of desirable phonemes and produce various combinations, but by and large they are not sensitive to the pronounceability, distinctness, or resemblance to the brand category of the names they output. Perhaps the success of the PNP classifier presented in Chapter 2 stems from its ability to capture the implicit sound-symbolic processes that influence brand creation and acceptance. Branding consultants are keenly aware of the "linguistic landscape" of competing names (Leben 2003). The tension of creating a name that is simultaneously new and distinctive while recognizable and interpretable within a category is borne out in the shaping of many of the PNPs in our experiments. If all names were really so unique, classification by category (e.g., identifying all drug names) would surely be an impossible task. The fact that drug names remain so recognizable is a sign that group familiarity is, at least for now, a dominant factor in the human generation of new names.

[15] See http://www.shinolas.com/blog/main.asp (visited: 5/5/2003)

Chapter 7:
Conclusions

The primary argument of this thesis is that word-internal character sequences provide surprisingly strong evidence for predicting the semantic category of proper names. This information is complementary to contextual cues present in surrounding text, and can be used independently or as an enhancement to a context-based classification system. By creating a statistical model of PNP generation, we have demonstrated that the inherent sound-symbolism of proper names can be captured and exploited to aid classification in a wide variety of domains, and even in multiple languages, achieving over 90% accuracy in many cases. The additional benefits of a machine-learning approach are that we can quickly acquire proficiency in new domains (requiring only lists of names in each category), and we can use the same model to generate novel proper names that mimic a given category. We have shown that these models can be extended to perform segmentation in conjunction with classification, achieving state-of-the-art performance in named entity recognition across multiple languages. Even in the final context-rich sequence classification model, the switch from word-level features to character-level features decreased errors by 25%.

The source of this method's success is the non-arbitrary relationship between the composition of names and the entities they describe. There are rich and pervasive sound-symbolic processes at work in proper names, which people are mostly aware of at a subconscious level. One exception is professional brand-name creators, who are keenly aware of sound-symbolism and exploit it extensively to create names that have the right combination of sounds and evoked meanings. Our model is picking up on the sound-meaning regularities tacitly in the minds of name consumers (general language users), and actively wielded by name producers (brand consultants). In this sense, this research is of value both as a practical advance in statistical natural language processing technology, and as an empirical enquiry into the structure and content of the linguistic and cultural forces that shape the creation of new names.

References

Abney, S. (1991). Parsing By Chunks. In Berwick, R., Abney, S., & Tenny, C., editors, *Principle-Based Parsing*. Kluwer Academic Publishers.

Appelt, D., Hobbs, J., Bear, J., Israel, D., Kameyama, M., Kehler, A., Martin, D., Myers, K., & Tyson, M. (1995). SRI International FASTUS system: MUC-6 test results & analysis. In *Proceedings of the Sixth Message Understanding Conference*, pp. 237-248.

Ara'ujo, M., Navarro, G., & Ziviani, N. Large text searching allowing errors. In *Proceedings of WSP '97*, pp 2—20. Carleton University Press.

Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., & Yarowsky, D. (*eds*). (1999). *Natural Language Processing Using Very Large Corpora*. Kluwer Academic Publishers.

Chitashvili, R. J. & Baayen, R. H. (1993). Word Frequency Distributions. In G. Altmann & L. Hrebicek, L. (eds.), *Quantitative Text Analysis*, Wissenschaftlicher Verlag Trier, Trier, pp. 54—135.

Baluja, S., Mittal, V., & Sukthankar, R. (1999). Applying machine learning for high performance named-entity extraction. *Proceedings of the Conference of the Pacific Association for Computational Linguistics* (pp. 365-378).

Banko, M., & Brill, E., (2001). Scaling to Very Very Large Corpora for Natural Language Disambiguation. In *Proceedings of the Association for Computational Linguistics (ACL 2001)*.

Bazzy, I., & Glass, J. (2000). Modeling out-of-vocabulary words for robust speech recognition. In *Proceedings of ICSLP*, Beijing.

Beeferman, D. (1996). Stochastic language identifier. Unpublished manuscript, Carnegie Mellon University. http://www.dougb.com/ident.html

Begley, S. (2002). New ABCs of Branding. *The Wall Street Journal*, Marketplace, 8/26/2002. Available online at < http://www.cintara.com/name_images/CintaraWSJArticle.pdf> (Visited: 12/13/2002).

Bikel, D.M., Miller, S., Schwartz, R. & Weischedel, R. (1997). Nymble: a High-Performance Learning Name-finder. *Proc. ANLP-97,* pp. 194-201.

Bikel, D., Schwartz, R. & Weischedel, R. (1999). An Algorithm that Learns What's in a Name. *Machine Learning* 34: 211–231.

Bodenreider, O., & Zweigenbaum, P. (2000). Identifying proper names in parallel medical terminologies. In *Medical Infobahn for Europe – Proceedings of MIE2000 & GMDS2000*, pp. 443-447.

Borthwick, A.. (1999). A Maximum Entropy Approach to Named Entity Recognition. Ph.D. thesis, New York University.

Campbell, D. A., & Johnson, S. B. (1999). A Technique for Semantic Classification of Unknown Words Using UMLS Resources. In *Proceedings of AMIA '99 Annual Symposium* (American Medical Informatics Association), session on *Innovations in NLP*.

Cavnar, W. B. & Trenkle, J. M. (1994). Ngram Based Text Categorization. *Proceedings of the Third Annual Symposium on Document Analysis & Information Retrieval*, pp 161-169.

Charoenpornsawat, P., Kijsirikul, B., & Meknavin, S. (1998). Feature-Based Proper Name Identification in Thai. In *NCSEC-98 (The National Computer Science & Engineering Conference '98)*.

Collier, N., Nobata, C., & Tsujii, J. (2000). Comparison between Tagged Corpora for the Named Entity Task. In the Proceedings of the 18th International Conference on Computational Linguistics (COLING 2000). Saarbrucken, German. pp. 201-207.

Collins M. & Singer Y. (1999). Unsupervised models for named entity classification. In *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing & Very Large Corpora*, pp. 189—196.

Cucerzan, S., & Yarowsky, D. (1999). Language independent named entity recognition combining morphological & contextual evidence. In *Proceedings of the Joint SIGDAT Conference on EMNLP & VLC*.

Cucerzan, S., & Yarowsky, D. (2002). Language Independent NER using a Unified Model of Internal and Contextual Evidence. In *Proceedings of the Sixth Conference on Natural Language Learning (CoNLL)*, Taipei, Taiwan.

Dunning, T. (1994). Statistical identification of language. Computing Research Laboratory technical memo MCCS 94-273, New Mexico State University, Las Cruces, New Mexico.

Fine, S., Singer, Y., & Tishby, N. (1998). The hierarchical hidden Markov model: Analysis and applications. *Machine Learning*, 32:41—62.

Florian, R. (2002). "Named Entity Recognition as a House of Cards: Classifier Stacking" In Proceedings of CoNLL'02, pp. 175—178, Taipei, Taiwan.

Freitag, D., & McCallum, A. (2000). "Information extraction with HMM structures learned by stochastic optimization". In *Proceedings of the American Association for Artificial Intelligence (AAAI-2000)*.

Freitag, D. & Kushmerick, N. (2000). Boosted Wrapper Induction. In *Proceedings of the American Association for Artificial Intelligence (AAAI-2000)*, pg. 577—583.

Hinton, L., Nichols, J., & Ohala, J. (*eds*.). (1994). *Sound symbolism*. Cambridge: Cambridge University Press.

Holdcroft, D. (1991). *Saussure: Signs, Systems & Arbitrariness*. Cambridge: Cambridge University Press.

Holman, C. (1997). TV on the edge: Idiot Box. *Creative Loafing, Atlanta, February 08 1997*. <http://www.creativeloafing.com/archives/atlanta/newsstand/020897/b_edge.htm> (Visited: 3/20/02).

Kahn, J. (2001). Man's search for meaninglessness. *Fortune, October 2001*. Available online at <http://www.business2.com/articles/mag/0,1640,16905,FF.html> (visited 5/5/2003).

Kazama, J., Makino, T., Ohta, Y., & Tsujii, J. (2002). Tuning Support Vector Machines for Biomedical Named Entity Recognition. In the Proceedings of the Natural Language Processing in the Biomedical Domain (ACL 2002). Philadelphia, PA,

Klink, R. R. (2001). Creating meaningful new brand names: A study of semantics & sound symbolism. *Journal of Marketing: Theory & Practice*, Vol. 9, No. 2, Winter 2001, pp. 38-45.

Krauskopf, L. (2002). Naming new drugs: Costly, complex. *The New Jersey Record, January 15 2002*. <http://home.cwru.edu/activism/READ/Bergen011502.html> (Visited: 03/20/02).

Kushmerick, N. (2000). Wrapper Induction: Efficiency & Expressiveness, Artificial Intelligence, pg. 118.

Leben, W. (2003). Personal correspondence.

Marcus, M., Kim, G., Marcinkiewicz, M. A., MacIntyre, R., Bies, A., Ferguson, M., Katz, K., & Schasberger, B. (1993). The Penn Treebank. *Computational Linguistics* 19:313—330.

McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction & segmentation. In *ICML-2000*.

Mikheev A., Moens M. & Grover C. (1999) Named Entity recognition without gazetteers. In *Proceedings of the Annual Meeting of the European Association for Computational Linguistics* (EACL'99), Bergen, Norway, pp. 1-8.

Mikheev, A. (1997). Automatic Rule Induction for Unknown Word Guessing. *Computational Linguistics* vol 23(3), ACL 1997. pp. 405-423.

Mikheev, A., Grover, C., & Moens, M. (1998) Description of the LTG System Used for MUC-7. *MUC-7*. Fairfax, Virginia.

Muslea, I, Minton, S. & Knoblock, C. (1999). A Hierarchical Approach to Wrapper Induction.

Nobata, C., Collier, N., & Tsujii, J. (1999). Automatic Term Identification & Classification in Biology Texts. In the Proceedings of the fifth Natural Language Processing Pacific Rim Symposium (NLPRS). Beijin, China. pp. 369--374.

Nobata, C., Collier, N., & Tsujii, J. (2000). Comparison between Tagged Corpora for the Named Entity Task. In the Proceedings of ACL 2000 Workshop on Comparing Corpora. Hong Kong, China. pp. 20-27. [PDF]

Ohta, T., Tateisi, Y., Mima, H., & Tsujii, J. (2002). GENIA Corpus: an Annotated Research Abstract Corpus in Molecular Biology Domain. In the Proceedings of he Human Language Technology Conference (HLT 2002).

Ramshaw, L. A. & Marcus, M. P. (1995). Text chunking using transformation-based learning. In Yarowsky, D. & Church, K., editors, *Proceedings of the Third Workshop on Very Large Corpora.*

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *EMNLP 1*, pages 133--142.

Riloff, E. (1996). Automatically Generating Extraction Patterns from Untagged Text. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pp. 1044—1049.

Rhodes, R. (1994). Aural images. In Sound Symbolism (Hinton et al. eds.), pp. 276—292.

Ruhlen, M. (1996). *The Origin of Language: Tracing the Evolution of the Mother Tongue*. John Wiley & Sons.

Saussure, F. de. ([1916] 1974): *Course in General Linguistics* (trans. Wade Baskin). London: Fontana/Collins.

Sereno, J. A. (1994). Phonosyntactis. In *Sound Symbolism* (Hinton et al. *eds*.), pp. 263—275.

Smarr, J., & Manning, C. (2002). Classifying unknown proper noun phrases without context. Technical Report dbpubs/2002-46, Stanford University, Stanford, CA.

Soderland, S.; Fisher, D.; Aseltine, J.; & Lehnert, W. CRYSTAL: Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pp. 1314—1319 1995.

Tjong Kim Sang, E., & De Meulder, F. (2003). Introduction to the CoNLL 2003 shared task: language independent named-entity recognition. In *Proceedings of CoNLL 2003*, pp. 142—147.

Wacholder, N., Ravin, Y., & Choi, M. (1997). Disambiguation of Proper Names in Text. In: *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pp. 202-208.

Weischedel, R., Meteer, M., Schwartz, R. Ramshaw, L., & Palmucci, J. (1993). Coping with ambiguity & unknown words through probabilistic models. *Computational Linguistics* 19(2), pp. 359—382.

Yorksten, E., & Menon, G. (2003). A sound idea: phonetic effects of brand names on consumer judgments. To appear in *Journal of Consumer Research*, June 2004. Available online at <http://pages.stern.nyu.edu/~gmenon/A%20Sound%20Idea.pdf> (Visited: 5/5/2003).

Viterbi, A. J. (1967). Error bounds for convolutional codes & an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* IT-13, pp. 1260—269.

Zipf, G. K. (1949). *Human Behavior & the Principle of Least Effort*. Cambridge: MA: Addison-Wesley.

Appendix

Size and source of each PNP data set

Category:	**drug** (6871 examples)
Description:	Micromedex 2000 USP Drug Index
Source:	my.webmd.com/drugs
Category:	**nyse** (3403 examples)
Description:	Companies on the NY Stock Exchange
Source:	www.nyse.com/listed
Category:	**movie** (8619 examples)
Description:	Internet Movie Database (IMDB) listing of 2000 movies and videos
Source:	us.imdb.com/Sections/Years/2000
Category:	**place** (4701 examples)
Description:	Collection of country, state/province, and city names from around the world
Source:	dir.yahoo.com/Regional/Countries
Category:	**person** (5282 examples)
Description:	List of people with online biographies
Source:	www.biography-center.com
Category:	**cheese** (599 examples)
Description:	Global database of cheese information
Source:	www.cheese.com
Category:	**disease** (1362 examples)
Description:	MeSH List of Diseases and Disorders
Source:	www.mic.ki.se/Diseases/alphalist.html
Category:	**band** (2234 examples)
Description:	Names of popular music bands
Source:	http://directory.google.com/Top/Arts/Music/Bands_and_Artists/
Category:	**artist** (1773 examples)
Description:	Names of popular musical artists
Source:	http://directory.google.com/Top/Arts/Music/Bands_and_Artists/
Category:	**car** (693 examples)
Description:	Automobile model names (modern cars)
Source:	http://autos.msn.com/compare/choose.aspx
Category:	**computer** (155 examples)
Description:	Names of desktop PC models
Source:	http://www.epinions.com/cmhd-Desktops-All-Pentium_III

The GENIA corpus is available at http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA

CoNLL 2003 shared task data is available at http://cnts.uia.ac.be/conll2003/ner

Answers to PNP challenge in Section 4.1.

PNP	**Category**
R & C	*Drug (for lice infections)*
Randall & Hopkirk	*Movie (1969 TV series)*
Steeple Aston	*Place (in Oxfordshire)*
Nandanar	*Person (Indian saint)*
Gerdau	*Company (Chilean steelmaker)*