Markup is Markup

David D. McDonald
Gensym Corporation
125 Cambridge Park Drive, Cambridge MA 02140
617 547-2500 x425, dmcdonald@gensym.com

Abstract

This position paper makes several simple points that locate the treatment of Web pages within the spectrum of approaches to the texts that are normally used by the linguistically sophisticated language processing community. The first point is that what ostensibly makes the Web a special kind of text, html markup and links, is just a variation, albeit a very populist one, on text types that much of the community has been working with for a long time. Two other points deal with specific techniques that the author has used in his own work with such texts which seem to be particularly effective at providing a graded syntactic and semantic analysis of unconstrained texts with markup.

1. The Impact of Markup

The most important point to make is that the truely significant difference is not between Web pages and other types of text but between texts that include markup, the Web among them, and texts that do not. All of our prior experience with markup of any sort carries over to the Web; we are not starting afresh. Taking the chance of restating the obvious, let me begin here by reviewing what constitutes markup, and then discussing some of the benefits and complications of doing linguistic analyses for content using texts with markup·

1.1 Different sorts of markup

Generally speaking, markup is the addition of information to a text that identifies its parts and their relationships to each other while not contributing to the text's wording or conventional content. The original notion is probably 'printer's marks'—character sequences that book publishers introduce into the electronic form of a manuscript to indicate to the typesetting devices when a region is to appear in bold, when the fonts shifts, alternations in leading, and so on. The idiosyncrasy and specificity of purpose of this sort of markup, however, make it not particularly well suited to nlp programs, something that is all too familiar to the people who have had to interpret and strip them from electronic dictionaries.

From the early days of professional computing we have marked up our documents with commands directed at formatting programs ranging from troff to Tex. This document itself is being written using invisible (wysiwyg) markup maintained by Microsoft's Word program. This markup can be made explicit, and thereby acces-sible by a program, by having the document written out using RTF, "rich text format" (as opposed to writing it out using one of Word's own custom formats or in 'plain ascii' or passing it through a postscript compiler for printing). The author has worked with texts (documentation) in ascii that incorporated standard markup derived by rule from the RTF version of the originals by a program written by Bran Boguraev. While RTF and its equivalents in the PC world are at least as awkward as printer's marks, they still are a means of having a document with markup as the starting point of one's analysis rather than just a stream of tokens and whitespace.

Standard markup in the form of sgml (html is just an instantiation of sgml) is far and away the simplest sort of markup to process (beause of the uniformity of its angle-bracket notation) while simultaneously being the most expressive (because it is an open-ended standard). In contrast with the other forms of markup, the thrust of sgml as a standard is to encourage the delimitation and naming of text regions (markup) according to semantic criteria rather than just as a means of constraining their presentation in some viewer.

In the NLP community, we have seen simple instantiations of semantic markup since the first LDC corpus distributions, and more recently with the advent of efforts to construct semantically tagged tree banks and with the results formats of MUC-6 (which were marked to delimit company and person names, dates, and so on). It is also beginning to appear on the Web. The SEC Edgar site with its data on corporate financial filings (www.sec.gov/edgarhp.htm), for example, looks odd to the uninitiated, since it comes up as plain text interspersed with notations in angle brackets. These notations are of course sgml tags, and they provide enough of a structural overlay to the text to permit it to be processed for content by very simple programs.

Unfortunately, even though sgml has the longer history and greater flexibility, html is what everyone has

· The work reported in this paper was done in 1993-95 while the author was self-employed. The opinons expressed here are his own and not necessarily those of Gensym Corp.

and is what virtually everyone uses. This can lead to the regrettable practice of encoding semantic information in patterns of html tags, a problem that will be touched on below.

There is no conflict in principle between having markup for both content and form. With the right software (which doesn't appear to exist commercially, though it would be nice to be shown wrong), the DTD of an sgml instantiation could be annotated with corresponding html tags. This would allow simple Web pages to be produced automatically from the sgml documents. (These pages would have no links or just those constructable by rule.) With adaptations to the browsing code we could then arrange for us or our programs to access the original, semantically tagged document files just as easily as we today can access the html files.

1.2 Focusing analysis

Here and in the rest of this paper I am taking the goal of the enterprise to be information extraction, which means the nlp analysis program will be answering questions such as what is this page about, or if that can't be determined, are there identifiable phrasal fragments that would give a better guess at the topic than an isolated word search; if the topic is one that we have a model for, what is this page saying. Vastly many other goals for applying NLP to the Web are conceivable and practical, but I will restrict the discussion here to just IE.[1]

Part of what makes the Web fascinating and important is that it is a populist medium where virtually anyone can be a content publisher, not just those with specialized technical skills or authors going through established, editorially-controlled channels (regular newspapers or magazines). This means that we have content from grade school kids and the SEC all on the same network. (I would venture that even the SEC would not have a substantial Web presence if it wasn't such an easy to do.)

Given that range, any IE application for the Web will have to deal with texts that aren't pre-selected by topic and will of necessity end up giving graded, partial results in most cases.[2] This is enormously difficult to do in a text that has no markup, but becomes more plausible as markup is added, and even more so to the extent that the markup indicates semantic rather than presentational types. The primary thing that this markup is adding is a capability to focus or narrow the analysis program's atten-tion region by region through the text.

A simple but quite serious case where focus is important is knowing when the text region being analyzed corresponds to a title. The motivation for this is that one of the more successful things on can do in the shallow semantic analysis of open texts is to identify which text sequences correspond to names. Given the name of a company, a person, a place, a movie, rock group, etc. that name can then be made available to a search engine, the page it is part of can be correlated with other pages making the same reference or constellation of references, both specifically or by type, or, given good techniques for following subsequent references, the sections of the page can be grouped to provide excerpts or a derived page produced with the names pulled out as links or the original page republished with the names marked up.

However, the standard technique for identifying new names depends on the capitalization of the words in the name as the means of delimiting it (see McDonald, 1993), and the capitalization convention in titles is completely different. If we know we are processing a title (e.g. we are within the title or h1 through h6 tags) we can turn off the capitalization-driven name finder and avoid substantial and confusing false positives.

Consider, for example, the title "Sears to Sell Mail-Order Line" (Wall Street Journal, 1/14/97). Assume that the analyzer has a reasonable lexicon for mergers & acquisitions and so will recognize the word "sell", leaving the segment "Mail-Order Line". The body of the article and our common-sense knowledge of the phrase 'mail order' reveal that this segment is a descriptive noun phrase rather than the name of a company ("... would sell its Fremans' mail order unit to ..."), but how is a name-finder application—which cannot, realistically, have a vocabulary large enough to include phrases for every type of product and service—know that that capitalized sequence isn't the equivalent of "North American Van Lines", which is a company.

When analyzing news articles (a non-trivial part of the Web), one can expect to see every person or company that is mentioned in a title repeated with their full names in the body of the article. Therefore if the markup will allow a title to be reliably distinguished from a body (and thereby ignored), the set of candidate names made available to other processes will be markedly more reliable.

Another example of markup providing focus is decoding pronouns within list structures. In the documentation I worked with (which originated as RTF and was then rendered into presentation-oriented sgml), a standard pattern was to introduce an operation within a paragraph

---

[1] I am also personally interested in generating pages dynamically. For instance in an application currently underway I am generating pages for the transcripts and requirements audits of undergraduates, which includes some pages that are generated on the fly in response to queries such as what courses are still available for satisfying a particular requirement given what courses from that group the student has already taken. In the future I expect to take up the problem of dynamically generating summaries.

[2] This happens to be precisely the application model for the original Tipster program: use powerful information retrieval to winnow the full set of texts into small sets by topic and then hand those texts off to semantically-specialized, sublanguage specific language compre-hension system (Information Extraction systems) that put the references and relationships that they have identified into database format for use by other reasoning or tabulating applications.

('putting a floppy in the drive slot'), and then in a sequence of list item segments, walk through all the steps that operation entailed. These steps tended to be phrased as imperative VPs, and tended to refer to the (linguistically) focused items with pronouns or very reduced DefNPs. By recognizing this pattern of markup, I was able to structure the search space for the references, recognize the leading words as verbs with great reliability, fill in the missing actor by default, and interpret the successive list items as a sequence of actions.

Note that this pattern of paragraph preceeding list items was not an explicit type in these document's markup. The markup was presentation-based like html is, and the pattern had to be encoded in the analyser's grammar just like any other sort of grammar rule. If the markup had been used for semantic purposes (which was not possible since it was derived by rule directly from RTF), then stipulating it in the grammar would be unnecessary since the semantic tag or tags could be directly treated as non-terminal categories.

Other instances of focus from markup are more direct. The structural markup that can appear in the header of a page (e.g. 'base') is often virtually a semantic tag and can allow one to directly apply a subgrammar to that segment or to apply a specific set of heuristics if the vocabulary in the segment is outside of what the analyzer knows (e.g. for addresses in signatures).

These examples of markup providing a focus to the linguistic analysis—selective processing modes and strong heuristic assumptions—were salient and important to the success of the projects in my own experience. I would expect there to be many more such examples in other people's work.

1.3 Processing markup

When we are treating a stream of characters as an ordinary natural language 'text'—which is what we want to do when we apply NLP techniques to Web pages—we have innocently assumed for decades that we could interpret all of the characters as just words, punctuation, or whitespace, and have built the lower levels of our analysis programs accordingly.

The introduction of markup complicates this picture considerably, and in this section I would like to touch on some of the techniques I have used in Sparser (McDonald 1992) to deal with these complications.

Parsing a set of sgml tags per se (<title> … </title>) is quite simple. It entails encoding the vocabulary of tags, anticipating the angle bracket patterns in that vocabulary (not every instance of an angle bracket pair is necessarily markup), and having a small grammar for organizing the pat-terns that can occur within the brackets (e.g. <input type="radio" name="6">). The question is what is done with those tag objects after they are recognized.

We would like the normal operation of the parser to be as unaffected by the presence of markup as possible, and we would especially like to be able to use our regular IE grammars with minimal or no modification. To achieve this, I found it important to distinguish between 'structural' tags (such as title, h1-6, or li) and what I call 'invisible' tags. Structural tags only occur at what we would otherwise think of as sentence or paragraph boundaries. They do not affect the interpretation of the sequence of tokens within the linguistic units that we feed into our sentence or phrasal grammars. The tag set in the Wall Street Journal corpus distributed by the LDC or in the Tipster corpus consists of just structural tags demarcating sentence and paragraph boundaries or indicating fixed data such as article id numbers, publishers, topic stamps and so on.

Invisible markup consists of such things as emphasis tags, anchors, and many of the instances of the break tag—generally speaking any tag structure (the angle bracket phrase taken as a whole) that disrupts the between-word adjacency relationships that drive linguistic analyses. From an IE system's point of view, invisible markup is whitespace with virtually no linguistic significance for the analysis, yet if it is handled like ordinary tokens it will cause the content analysis to fail because the parser will not see the expected adjacency patterns among the words in the non-markup vocabulary.

Rather than complicate the parsing algorithm to appreciate and skip over these tag structures, my approach in Sparser is to hide them (hence "invisible"). Sparser uses a chart that is implemented with objects that explicitly represent the 'positions' between words. One of the fields on a position records the whitespace (if any) that occurred between the word to the left of the position and the one to its right.

Intervening markup that has been indicated in the grammar as invisible is caught by the processing level between the low-level tokenizer and the process that introduces (regular) words and punctuation into the cells of the chart, and is placed in this whitespace slot in the intervening position object. Stored there, it is truly invisible to the chart-level parsing algorithms just as though it were ordinary whitespace, while still being available to processes that republish the text or that want to heuristically interpret the implications of, e.g., some phrase being emphasized.

2. Adapting IE to Unrestricted Documents

The usual application for an IE system is to process a preselected corpus of documents on a single topic in a single register (or a set of these), where a semantic model, target database representation, and sublanguage grammar have been prepared in advance specifically for that corpus.

The Web can deliver such documents as well or better than any other network medium given unambiguous structural markers on the target documents or a url with a guaranteed content, but that scenario does not fit 99% of what is actually out there, and we would still like some way of adapting what have proved to be quite successful

techniques from information extraction to this wider world of documents.

In the following two sections I would like to offer up some techniques I have used when applying Sparser to unrestricted texts. This work is anecdotal since the implementations were prototypes and the applications small, but they are offered in the hope that others will find them useful as well.

2.1 Two-level phrasal parsing with semantic grammars

When properly disciplined to conform to sound grammatical generalizations, semantic grammars are a quick and natural way to write an IE application. The results are nevertheless 'grammars' in the ordinary sense, and a parser that uses them will still be depending on getting an analysis for virtually all of the phrases and grammatical relations in its target sentences in order to arrive a correct description of their meanings.

Getting such thorough parses, however, depends on there being a relatively small amount of variation in how the documents of the corpus are phrased. For any fixed amount of development that has gone into the preparation of the grammar, we know that the greater the amount of variation there is in the corpus, the more likely it is that the grammar will have a gap in its coverage of a phrase type or a pattern of grammatical combination.[3]

What this comes to in practice is that one can do fairly well in a short amount of time if the corpus comes from a single, well edited source; but the coverage will be markedly worse with a long tail of small exceptions if the source varies from document to document. I recently was faced with just such a situation in a prototype application to extract corporate earnings information, and developed (what may be) a new technique of using semantic grammars to cope with it.

Initially, I wrote a full sentential-coverage semantic grammar targeted at a trivial sample (six short articles) of earnings reports taken from the Wall Street Journal. It took roughly a day to get 100% coverage on that sample, with most of the time going into working out a general model for 'change in amount' phrases such as "Net revenues for the six months ended June 30, 1996, increased 54% to $27,605,000 compared to $17,925,000 for the six months ended June 30, 1995." Testing on a (trivial) reserved set of two more articles from the Journal was then accurate in 7 out of 9 sentences (tuples) with the problems being a new case in the date phrase type and a new way to phrase the main relation type, which is not too bad as such things go, and was simple (ten minutes) to remedy.

Newspaper articles on earnings reports are worthless commercially, however, since they appear the day after the event and all the resulting stock trading has already happened. So the next step was to go to the press releases from which the articles were derived. These tend to appear on the Web at about the same time as they are submitted to PRNewsWire or its equivalent, so they are likely to actually be useful.

Unfortunately, the initial results on twenty 'ern' press releases using the given grammar and parsing algorithm were pitiful: three out of roughly forty possible tuples were correctly recovered. Several more hours of adding cases changed the results very little, since it turned out that literally every press release varied in one or more ways in how it patterned the elements of the earning relation (i.e. company, financial item (sales, earnings, etc.), amount, time period, percent change, reference time period, amount in that reference period). A quick examination by hand of an additional ten press releases suggested that the accumulation in variation was not going to top out with any small amount of additional work.

In retrospect, this result could have been anticipated from the fact that, unlike the articles from the Wall Street Journal, each of the press releases had a different author, each with his or her own particular style of phrasing. The gross patterning was the same, presumably because the authors were quite familiar with this genre of article, but seen in detail, as a parser does, each author used slightly different phrasings or orderings for one item or another, and the cumulative effect was to introduce small but fatal gaps in the grammar's coverage.

This is a telling result for applying an IE system of standard design to the Web, since on the Web, in contrast to the newspaper or newswire articles these systems were developed for, virtually all the documents have different authors. An alternative design is needed if these systems are to work on the Web in any general way.

In response to this problem, I developed a prototype of an alternative way of parsing the text in the press releases, one that could well have useful application in a wide range of text types provided that certain criteria are met.

(1) The target content should center primarily on phrases and less on clauses (e.g. mostly weak verbs).

(2) There should be relatively little variation at the phrasal level.

(3) There is a largely canonical order to how the relevant information is presented.

[3] The results of the last several MUC evaluations have reached what some people have called a 'glass ceiling' in that the performance of all of the best systems seems to have remained constant at roughly fifty percent recall/precision for several years in a row. Having looked at the MUC corpora and done similar work myself, I conjecture that the reason the results are so bad and show no sign of improvement is that the range of variations exhibited in the training corpora is only a fraction of what appears in the genre as a whole (and that assumes there was enough time to examine the whole training set, which is often not the case). When the test corpus is run, the odds of getting patterns that were not anticipated is consequently quite high, and as a result today's IE techniques for semantic analysis suffer since they are largely dependent on coding the patterns directly.

This description appears to cover meeting announcements, person-company-title relations, addresses, possibly things that anchor off of dates (the beginning of the page with the call for this symposium falls into that category, with its times for registration information, the plenary session, due date, etc.), not to mention a vast number of specialized, commercially important, mundane announcements.

In such a domain and register where the variation within the content-carrying phrases is quite small while the variation in the grammatical 'glue' that links them together is relatively large (because of variation in syntactic relations and simple lexical choices like prepositions), the idea is to ignore the glue and only tally up the phrases and their gross order. Doing this entails running a conventional, phrase-oriented partial parser over the text using a semantic grammar, and then feeding its results, phrase by phrase, to another level of surface analysis, ignoring the glue or unrecognized phrases between the recognized ones.

This second analyzer is just a transition net, where the transitions are conditioned by the categories of whole phrases, with no assumption that successive phrases are adjacent. (This is equivalent to a conventional recursive transition network parser with wildcard transitions between each state. A two-level system is just a simpler way to implement such a ubiquitous distribution of wildcard transitions.) Here, for example, is the network that I wrote for the earnings press releases. The abbreviations labeling the arcs are C for company, Q for quarterly time period ("the first quarter of 1996"), Y for yearly period, FD for financial datum ("sales", "earnings", etc.), and $ for an amount of money or $/s for an amount per share. It starts running at the beginning of the press release. The Continue state is both the accept state and a hook for optionally going on to look for additional relations later in the article.

The goal of this network is to find the first (and invariably the most important) earnings report relation in the press release. Somewhat surprisingly, it correctly identified all of those relations in the small set of press releases; but of course this is a case of testing on the training corpus, a well known fault that makes any result suspect. Notably, it failed when I hand simulated it on the very first earnings press release I saw while preparing for this paper. It does not recognize the sequence FD - Q, as in "Earnings for the fourth quarter, 1996", which was the dominating pattern of such information in that article.

This is only an anecdotal result, since it was run on a minuscule corpus as part of a very short-lived project, and additional complexity would certainly be needed for broad, robust performance. We can get a glimpse of some of that complexity by considering the root of the problem with the failure to get "Earnings for the fourth quarter, 1996". The problem is the assumption, built into this initial design, that the phrasal constituents will appear in a fixed order. It is a major failing, and would only be exacerbated if we moved from English to a language with free phrase order where we could no longer even depend on syntax to stipulate most of the ordering.

The motivation behind the use of a fixed order was to avoid false positives (crucial in any commercial application). In the earnings reports genre, virtually every sentence that announces new earnings figures will include a trailing adjunct that gives the results for the comparative time period. The connectives that indicate that a comparison is being made are one of the elements of this genre that varies widely among different authors, while the phrases in the adjunct are virtually identical to those in the primary announcement clause. Since this means that we want to avoid needing to know anything about the connectives, insisting on seeing the phrases just once and in the stipulated order is a way to reliably get the positive announcement while avoiding the comparison.

The obvious fix in this particular example is to create two different paths to the state that currently follows FD, one for each of the orderings of financial-item ('earnings') and time-period. This style of fix (add more paths), when extended to its logical conclusion, takes us out of the domain of transition networks into a 'check list' style of analysis, where the ordering of the phrases is less indicative than just their presence. That is a quite different algorithm, but a reasonable one to explore, and it keeps with the theme of this line of work: emphasise the relative uniformity of the within-phrase patterns as compared with the clausal and sentential patterns, and look for patterns in the association of phrases that are reliably indicative of the relations we would have found if we had been able to do a full analysis of the entire text.

The greatest problem in any phrase-only approach is accurately bounding the region of text within the phrases of the pattern have to occur—a problem that never arises in a conventional full analysis. There are shallow techniques for accurately identifying sentence boundaries with only minimal lexicons (principally of abbreviations) that might be sufficient, but this remains to be explored.

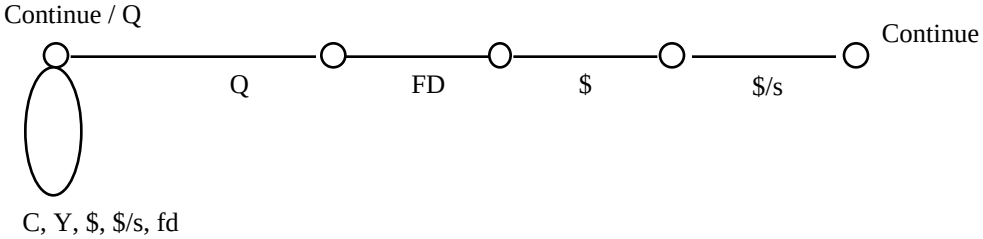

Figure 1: A network for extracting earnings report data

2.2 Alternatives to POS

Given a semantic grammar, there is no particular need for knowing the syntactic part of speech (POS) of a word since its role(s) in the grammar will be dictated by the semantic category(ies) it falls into rather than its syntactic categories. On the other hand, the vast bulk of the text one would like to process on the Web and elsewhere is in subject domains for which one does not have a semantic grammar available and should not expect to have it given the considerable labor and skills involved in writing such grammars and their attendant conceptual models.

That fact not withstanding, there is a significant amount of linguistic 'value-added' that can be brought to the handling of a text without a semantic model of the domain of information the text is in—we should be able to do better than just viewing the text as a bag of potential keywords accessible from a search engine.

In particular, if we can reliably parse the text into minimal phrases, we have the beginning of a means of bootstrapping a default semantic model of the subject by making shallow semantic interpretations of the syntactic relationships within the phrases: head words are taken as kind terms, modifiers as indicating subtypes, common modifiers like "new" or "another" providing views on the objects, and so on. (And if we can do a clause-level parse on top of those phrases with some reliability, then this lets us establish relations between kinds and the actions over them: 'who does what to whom'.)

With such a model, we can accurately differentiate or link segments of the text to provide derived views or to aggregate related parts of different texts, and, given some heuristics, we can start to identify segments as introductions of new subjects versus elaborations of established ones, which will aid in navigation and in the sorting and synopsizing of the enormous number of articles returned by broad keyword searches.

Given the proliferation of reasonably accurate statistical part-of-speech programs in recent years (e.g. Brill 1992, Voutilainen et al. 1992, Weischedel et al. 1993), it has come to be taken for granted that the way one identifies phrases is by applying one of these programs to the text in order to label each word with its POS (sometimes rank-ordered when there are alternatives), and to then run a trivial parser over the sequences of labels, noticing sequences like Adj+Adj+Noun+Noun and taking those to be the phrases.

The trouble, of course, is that the phrases are only as accurate as the underlying statistical n-gram model of POS labeling, and this is subject to two problems. One is that when the program is reported to have an accuracy rate of, e.g., 95% that means that it will make one mistake in every twenty words, and more to the point, that mistake will be on a word that is important for the phrase determination: a noun mislabeled as a verb or visa versa for example; the mistake will never be in the identification of a function word. (Words like "that" or "to", which can serve different grammatical functions, are an exception, and there can be disagreement as to what label other function words should have.)

The other, more subtle problem is that any statistical algorithm depends for it accuracy on the amount, and also the genre, of the text it is trained on. I had the experience of working with a text that was tagged by a version of Eric Brill's POS analyzer (circa late 1994), that yielded dramatically bad results—the reason being that it had been trained on conventional texts but was being applied to software HelpDesk reports, and as a result did very badly with file names, operating system versions, embedded code, and the like, which of course had not been part of its training text and consequently one would naturally expect that it might not do well.

I have developed an alternative method for separating word sequences into phrases that, in the one case where I have been able to make a direct comparison, did just as well as a very well regarded statistical POS program (which for contractual reasons must remain nameless), and did this without requiring any training whatsoever. The genre was software documentation and the task was index construction and the detection of actor-action relationships in order to seed a Help system.

This method simply uses function words and productive morphology plus a very small state machine. That sentence, for example, plainly starts with a noun phrase as indicated by the word "this", and the NP is just two words long because the third word ends in +ly and so must be an adverb. The presence of a preceding adverb provides a disambiguator for the 'ends-in-s' word that follows ("uses") which forces the word to be taken as a verb rather than a plural noun. A robust heuristic about verb groups says that the only time they extend beyond a content word head (here "uses") is with marked adverbs; the absence of which provides a motivation for declaring that the next phrase starts with the word "function". The "and" two words later provides a reason to end that phrase.

In this method, one starts with a set of annotations on all the closed class words in the system's vocabulary which indicate whether a word closes a phrase just before it, indicates that a phrase begins just after it ("and" does both), or itself initiates the beginning of a phrase (e.g. articles, auxiliary verbs). These annotations are appreciated during the scan phase of the parse, and boundaries drawn accordingly. The mean distance between closed class words or boundary-inducing markup in the documentation corpus was 3.2 words.

The standard class of mistakes this algorithm makes come from having overly long phrases at those points in the grammar of a sentence where the transition between phrase is frequently not accompanied by a function word. In the example sentence, the (single) mistake is to include the word "plus" with the phrase just before it to yield "productive morphology plus". This is a subject / verb-group boundary; verb-group / object is the other major site that is prone to occurring without a definitive marker.

We can improve on the situation by seeding the system's vocabulary with a list of words that are known to

only occur as verbs and those that are either verbs or some other part of speech. (Note that we don't need all of the verb spelling forms since we get the regular past and progressive (ing) form for free from their morphology. Of course that still leaves the main verb vs. participle ambiguity.)

We adopt the heuristic that if the word can be a verb then it is unless we have explicit evidence to the contrary. A typical high frequency case is the spelling form "use", which the system takes to be a verb unless it appears in a context such as "the use" or "use of", in which case the state machine that reads the annotations and inserts the boundaries will overrule the default assumption. Given that boost, the performance of this algorithm in assigning phrase boundaries in the documentation corpus was indistinguishable from that of the statistical POS algorithm in the several sections of documentation that were checked.

To reiterate, the utility of this alternative algorithm for establishing phrase boundaries is that it does not require training. Training is always expensive. (Who would fund the construction of an annotated training corpus for alt.sex?) Furthermore, in a statistical system the performance on unknown words (lowercase content words that did not appear in the training set) is degraded compared to known words (it drops back to the a priori probability of noun vs. verb. vs. adjective in the same context). This algorithm, on the other hand, can treat virtually all the content words as unknown and still perform at a level that is not shabby. (In the example above, missing the word "plus", made it effectively was accurate to one word in fifteen). And it needs only dictionary POS data about verbs to perform markedly better.

I would venture to say that on the Web we will see a quite significant and ever growing body of words that are 'unknown' when compared to the vocabulary in, e.g., the Penn TreeBank, and that this will be an issue as we select shallow linguistic algorithms for processing it.

3. Summary

In this position paper I have tried to make a few simple points. The first is that there is nothing special about html and Web pages per se, rather it is the fact that these texts contain markup information—a variation on sgml—and that this is the property we should be focusing on when we adapt the text-handling algorithms of our parsers to the Web. By considering markup of all sorts when we modify our designs, we will retain the flexibility to handle semantically-loaded markup information once it becomes more common, and will keep presentation-oriented markup in the correct perspective as a separate kind of information. When presentation markup is pressed into service to indicate semantic facts, we will know to keep our encoding of such tag patterns as separate parts of our grammars rather than as intrinsic facts about the tags.

As part of the mechanics of handling tags within a parser, I have pointed to the difference between tags that label structural components of a document and those that modify or annotate text segments to indicate how they are to be presented or that they have been overloaded to indicate hyperlinks (, <a>). By treating modifications and annotations as whitespace that is invisible to the flow of the text content, we can use the content analysis aspects of our parsers without any modifications at all.

state of the art for coping with variations in wording is such that we much settle for partial, graded results, and apply a judiciously choosen set of interpretation heuristics. We cannot expect the fully analyzed, totally confirmed results that we can achieve in the single topic, editorially uniform domains like those used in the MUC conferences (e.g. short news articles announcing joint ventures or executives changing their jobs).

I have suggested that the minimal phrase types in any domain are an important anchor point on which we could build a heuristic IE analysis (names, descriptive noun phrases, standard adverbials such as dates or times, amounts, etc.). The two-stage semantic grammar described here has worked surprisingly well in one domain where the text style is highly stylized and the NPs carry virtually all of the information of interest. It has definite flaws, but nevertheless it is suggestive of a way to approach the problem of how to do shallow, heuristic, semantic analysis for information extraction on corpora where the range of variat Regarding content, I have pointed out that if our goal is information extraction and we are going to address anything like the full diversity of authors, topics, and genres that the Web presents us with (rather than do business as usual with single fixed domains and genres), then we are going to have to look for new techniques that will allow us to begin to cope with the enormous range of variations in phrasing and content selection that we will face.

There will be some regions of the Web where the information is sufficiently simple and its presentation sufficiently stylized and uniform that we will be able to write standard IE grammars for them, especially if we include markup tags within the grammar. Simple reports of tabular data such as commodity prices are example.

For everything else, the ion is too large to handle with a conventional inform-ation extraction engine.

When there is not even a grammar of simple phrases to start with, it is still possible to use the linguistic knowledge that is at the core of many of our NLP systems to do a better job of analyzing a page than keyword search engines, since we can use that knowledge to segment the text into phrases on purely syntactic grounds. Those phrases can then be used to improve the search, to allow the text to be organized into sections by topic continuity, to provide good indexes automatically, or even to provide the basis for a semantic domain model once the relationships implied by the organization of the words within the phrases has been reviewed and augmented by a person.

While the customary way to decompose a text into minimal phrases is to use a statistical part of speech (POS) analyzer, I reminded the reader that the results from such systems are only as good as the training data they have

received. Annotating texts by hand to create this data is a grueling task, however, and one that is not done cheaply. The enormous range of genres and subject matter on the Web make it unlikely that the data needed for an accurate assignment of POS will always be available, with an attendant degredation in the performance of statistical systems.

I have described an alternative scheme for identifying the phrases in a text that does not require training, just the construction of a simple grammar of how the close-class words of the language indicate where phrases begin and end. Just using that information alone, this algorithm runs at better than 80% accuracy. (Though it should be noted that this was measured on professional documentation, performance on markedly different genres (chat rooms or mailing lists) may be quite different.)

If the knowledge of close class words is augmented with information that can be obtained from a dictionary about whether a word is always or sometimes a verb, then the performance of this alternative, non-statistical algorithm for delimiting phrases was indistinguishable from that of a well-regarded statistical system. This being the case, I believe it holds significant promise, and might go a long way towards allowing us a foothold in the application of strong, linguistically informed natural language processing techniques to the World Wide Web.

4. References

Brill, E. (1992) "A simple rule-based part of speech tagger" Proc. Third Conference on Applied Natural Language Processing, ACL, Trento, Italy.

McDonald, D. (1992) "Robust Partial-Parsing through Incremental, Multi-level Processing" in Jacobs (ed.) *Text-based Intelligent Systems*, Lawrence Erlbaum, pp. 83-99.

McDonald, D. (1993) "Internal and External Evidence in the Identification and Semantic Categorization of Proper Names" Proc. ACL Workshop on Acquisition of Lexical Know-ledge from Text, June 21, 1993, pp. 32-43.

Voutilainen, A.; Heikkila, J.; and Anttila, A. (1992) "Constraint Grammar of English", Pub. No. 21, Univ. of Helsinki, Dept. of General Linguistics.

Weischedel, R; Meteer, M.; Schwartz, R; Ramshaw, L.; and Palmucci, J. (1993) "Coping with ambiguity and unknown words through probabilistic models" *Computational Linguistics.*