Understanding Captions in Biomedical Publications

William W. Cohen
william@wcohen.com
CALD
Carnegie-Mellon University
Pittsburgh PA 15213

Richard Wang
rcwang@andrew.cmu.edu
Dept. of Computer Science
Carnegie-Mellon University
Pittsburgh PA 15213

Robert F. Murphy
murphy@cmu.edu
Dept. of Biological Sciences & CALD
Carnegie-Mellon University
Pittsburgh PA 15213

Abstract

From the standpoint of the automated extraction of scientific knowledge, an important but little-studied part of scientific publications are the figures and accompanying captions. Captions are dense in information, but also contain many extra-grammatical constructs, making them awkward to process with standard information extraction methods. We propose a scheme for "understanding" captions in biomedical publications by extracting and classifying "image pointers" (references to the accompanying image). We evaluate a number of automated methods for this task, including hand-coded methods, methods based on existing learning techniques, and methods based on novel learning techniques. The best of these methods leads to a usefully accurate tool for caption-understanding, with both recall and precision in excess of 94% on the most important single class in a combined extraction/classification task.

1 Introduction

The vast size of the biomedical literature makes it essential to summarize pertinent scientific results. Normally this is done by creating curated databases, like the Entrez databases, SwissProt, and YPD. However, curated databases are expensive to create and maintain; do not typically permit extensive links to specific supporting data; do not estimate confidence of assertions; do not allow for divergence of opinion; and do not readily permit updating or reinterpretation of previously entered information. Information extraction (IE) methods can be used to at least partially overcome these limitations by automatically extracting information from biomedical text [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11].

Most previous biomedical IE systems have been applied to abstracts. Abstracts are a natural source of information, as they are readily available, and dense in information. A second "information-dense" part of scientific publications are the *figures* and *accompanying captions*. In most genres of scientific publication, the most important results in a paper are illustrated in non-textual forms, such as images and graphs. Authors understand that figures occupy large amounts of valuable page space, and are likely to be seen disproportionately by casual readers. Thus figure captions often concisely summarize a paper's most important results as perceived by the author.

However, applying IE to caption text is also problemmatic. Since the main purpose of caption text is to comment on an image, captions are often littered with references to the image, and these "image pointers" are interspersed with grammatical text in a variety of ways. This sort of extra-grammatical structure is likely to mislead automated extraction tools that assume grammaticality (either explicitly, by using NLP components such as part-of-speech tagging and shallow parsing, or implicitly, by virtue of being tuned on grammatical corpora). Detecting and understanding such

Fig. 1. Kinase inactive Plk inhibits Golgi fragmentation by mitotic cytosol. (A) NRK cells were grown on coverslips and treated with 2mMthymidine for 8 to 14 h. Cells were subsequently permeabilized with digitonin, washed with 1M KCl-containing buffer, and incubated with either 7 mgyml interphase cytosol (IE), 7mgyml mitotic extract (ME), or mitotic extract to which 20 mgyml kinase inactive Plk (ME + Plk-KD) was added. After a 60-min incubation at 32C, cells were fixed and stained with anti-mannosidase II antibody to visualize the Golgi apparatus by fluorescence microscopy. (B) Percentage of cells with fragmented Golgi after incubation with mitotic extract (ME) in the absence or the presence of kinase inactive Plk (ME + Plk-KD). The histogram represents the average of four independent experiments.

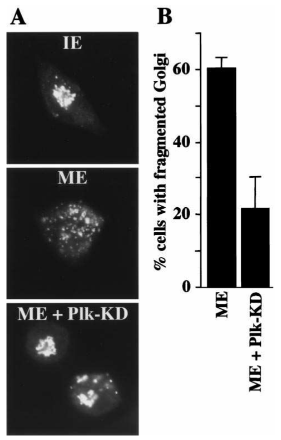


Figure 1: A figure reproduced from the biomedical literature. The caption includes both "citation-style" and "bullet-style" image pointers.

extra-grammatical structures would be beneficial before applying "off the shelf" IE components to caption text.

A second motivation for wanting to "understand" the structure of captions is to leverage systems that extract information from the *images* in scientific publications. In previous work, we have developed tools to automatically analyze fluorescence microscope images of cells and compute features relating to subcellular localization [12, 13, 14]. These tools have also been applied to images harvested from online biomedical publications [15], and we are currently extending this system to extract assertions such as "Figure N depicts a localization of type L for protein P in cell type C". This requires extracting protein names and cell names from captions and determining what microscope image each entity name refers to. Figures which contain multiple microscope images (the vast majority!) this task requires caption understanding.

In this paper, we will address the question of "understanding" the structure of caption text in a fairly general context. Our main goals are first, to detect extra-grammatical constructs in caption text, in particular extra-grammatical insertions that refer to parts of the accompanying image; and second, to determine what parts of the caption text are associated with which (references to) parts of the image.

2 The Caption-Understanding Problem

2.1 Motivating examples

Figure 1 illustrates some of the key technical issues in understanding captions. In the figure the boxed area encloses a prototypical figure harvested from a biomedical publication,[1] with the italicized text ("*Fig. 1. Kinase inactive Plk...*") being the associated caption from the reproduced figure. The reproduced caption text contains several strings that refer to places in the accompanying image: "A", "B", "IE", "ME", and "ME+Plk-KD" (the last two of which occur twice). Below we will call these strings *image pointers*.

[1]This figure is reproduced from the article "Ras Regulates the Polarity of the Yeast Actin Cytoskeleton through the Stress Response Pathway", by Jackson Ho and Anthony Bretscher, *Molecular Biology of the Cell* Vol. 12, pp. 1541–1555, June 2001.

There are at least two possible end goals for caption processing. One is to convert the caption text to ordinary grammatical text. Generally, one would like to *identify and remove* image-pointer related disfluencies, and recover any formatting or structural information lost as a consequence of the text's appearance as a caption. In the specific case of Figure 1, this would mean replacing the strings "(A)" and "(B)" with paragraph breaks, and removing all the other image pointers.

A second possible goal is to *associate subsequences of caption text with specific image pointers* and therefore, to parts of the image. If one imagines applying an IE system that utilizes image-processing to Figure 1, for instance, it might be useful for the system to know that all three microscope images are of NRK cells stained with anti-mannosidase II antibody, but that only the panel labeled "IE" concerns interphase cytosol.

Some additional issues that arise in caption-understanding are not illustrated by Figure 1. In Figure 1, all parenthesized expressions are image pointers, and vice-versa; however, this is not generally the case. Captions can also contain image-pointer strings that are not grammatically null, as in the text "Following a procedure similar to that used in (A), cells were stained for...".

2.2 Caption-understanding as extraction and classification

We decided to break down the caption-understanding task into several subtasks.

The first step is *image-pointer extraction*—identifying all image pointers in the caption. After image pointers are identified, they are *classified* according to their linguistic function.

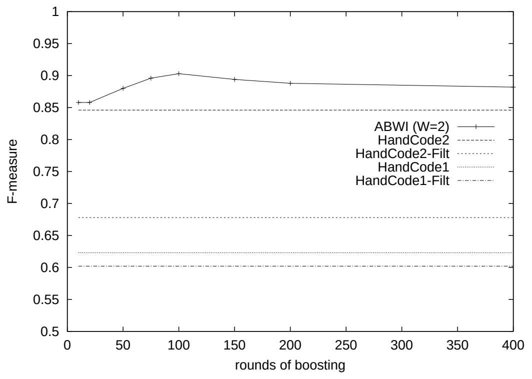
- *Bullet-style* image pointers function as compressed versions of bulleted lists. The strings "(A)" and "(B)" in Figure 1 are bullet-style image pointers.
- *NP-style* image pointers are used as proper names in grammatical text. An example is the string "(A)" in the text: "Following a procedure similar to that used in (A), ..."
- *Citation-style* image pointers are interspersed with grammatical caption text, in the same manner that bibliography citations are interspersed with ordinary text. The remaining image pointers in Figure 1 are citation-style.

After image-pointer classification, the *scope* of each image pointer is determined. The scope of an image pointer specifies, indirectly, what text should be associated with that image pointer. The scope of a NP-style image pointer is the set of words that (grammatically) modify the proper noun it serves as. The scope of a bullet-style image pointer is all the text between it and the next bullet-style image pointer. The scope of a citation-style image pointer is some sequence of tokens around the image pointer, usually corresponding to a nearby noun phrase.

As an example, applying these rules to the caption of Figure 1 produces these results (among others): the text "NRK cells" is associated with image pointer "A"; the text "fixed and stained with anti-mannosidase II antibody" is associated with the image pointers "A"; and the text "mitotic extract" is associated with "A" and the first occurence of "ME", but not with "IE" or "ME+Plk-KD".

In this paper we will focus on the first two steps, of *image-pointer extraction* and *image-pointer classification*. We will not discuss image-pointer scoping at any length. However, the scoping rules for bullet-style image pointers are trivial to implement, and NP-style image pointers appear to be rare. We believe that a reasonable approximation to scoping for citation-style image pointers can be implemented by using NP-chunking techiques [16, 17].

Method	W	Precision	Recall	F
HandCode1		98.5	45.6	62.3
HandCode1-Filt		99.2	43.2	60.2
HandCode2		74.5	98.0	84.6
HandCode2-Filt		89.0	54.8	67.8
aBwi	1	82.9	92.8	87.6
aBwi	2	89.7	91.0	**90.3**
aBwi	3	90.6	83.4	86.9
NrBwi (Slipper)	3	96.09	85.17	**90.3**
NrBwi (Ripper)∗	3	88.08	87.15	87.6
NrBwi (SVM)	3	100.00	75.20	85.6
NrBwi (SVM)∗	3	69.01	78.00	73.2



	W=2			W=3		
Method	Precision	Recall	F-measure	Precision	Recall	F-measure
aBwi	89.7	91.0	90.3	90.6	83.4	86.9
aBwi/na	85.2	91.6	88.3	85.9	92.2	89.0
saBwi/na	85.7	92.9	89.2	88.6	93.8	**91.1**

Figure 2: Top left: precision, recall and F-measure on image pointer extraction for a number of different. Top right: sensitivity of aBwi (W=2) to rounds of boosting T. Bottom: Performance of standard aBwi with and without engineered features, and the symmetric version (saBwi/na) with engineered features.

3 Experimental Results

3.1 Data Collection

To evaluate approaches to caption-understanding we collected a dataset of 100 biomedical publications in PDF format. This was a random subsample of a larger set of Pubmed Central papers previously used to evaluate a system (now called Slif[2]) for processing fluorescence microscope images [15]. Slif used a modified version of Pdf2Html (a public domain tool) to extract figures and captions from PDF documents, and experiments showed this system to have high precision (98%) and reasonable recall (77%) on this task. Using this tool, some figures were extracted from 90 of the 100 sample papers. We randomly chose one figure from each successfully-processed paper, for a total of 90 figures, and hand-labeled all the image pointers in the figure captions with an interactive tool. There are a total of 562 image pointers, or an average of 6.2 per caption. All but five of the figures contain at least one image pointer.

3.2 Hand-coded extraction methods

As a baseline, we first hand-coded some relatively simple extraction methods. Many image pointers are single letters (like "A" and "B") or simple variants. The HandCode1 method thus extracts as image pointers all expressions of the form "(X)", "(x)", "(X-Y)", "(x-y)", "(X and y)" or "(x and y)", where "X" and "Y" (respectively "x" and "y") indicate any uppercase (respectively lowercase) letter. In the patterns, any number of extra spaces are also allowed around each letter. HandCode1 has high precision (98.5% on our dataset), but low recall (45.6%).

[2]For "Subcellular Location Image Finder."

Most image pointers are parenthesized. The HandCode2 method thus extracts all parenthesized expressions that are (a) less than 40 characters long and (b) do not contain a nested parenthesized expression, and also extracts all whitespace-surrounded expressions of the form "x", "X", "x-y" or "X-Y" that are preceeded by one of the words "in", "from", or "panel". HandCode2 has high recall (98%) but only moderate precision (74.5%) .

Image pointers are often based around a sequence of upper- or lower-case letters from the beginning of the alphabet. HandCode2-Filt attempts to exploit this regularity. First, all HandCode2-generated expressions are collected and *normalized* by replacing X-Y constructs with the equivalent complete sequence. (*E.g.*, constructs like "B-F" are replaced with "B,C,D,E,F".) The the longest initial alphabet sequence "a,b,c,..." or "A,B,C,..." such that *every* letter in the sequence appears in some normalized HandCode2 expression is computed. (For instance, in Figure 1, this sequence would be "A,B", since "C" is the first uppercase letter not appearing in a parenthesized expression, and "a" is the first lowercase letter not appearing in a parenthesized expression.) Finally all HandCode2 expressions *not* containing a letter from this longest sequence are discarded. This filtering step improves precision, but unfortunately has a high cost in recall.

We evaluated these methods on the labeled data, and summarized the results in the first table of Figure 2. (The HandCode1-Filt method uses the same filtering heuristic as HandCode2-Filt to filter expressions from HandCode1.) In addition to precision and recall we also show the F-measure (the geometric mean of recall and precision) for each method. The hand-coded methods work only moderately well for this task: while it is easy to obtain a recall of 40-50% with very high precision, it appears difficult to identify the remaining image pointers with high precision.

3.3 Learning methods

3.3.1 Background.

There is a substantial literature using machine learning approaches for difficult extraction problems. One simple but successful extraction method is *Boosted Wrapper Induction (*Bwi*)* [18]. Bwi inputs on a tokenized version of a document, annotated with a set of *target strings* (strings to be extracted). From this data, Bwi learns three classifiers: the *fore classifier* identifies tokens that begin a target string, the *aft classifier* identifies tokens that end a target string, and the *length classifier* identifies lengths (*i.e.* 1,2,3,...) that correspond to target strings. All of these classifiers output a *confidence* or score. The aggregate score of the string between token i and token j is then $FORE(i) \cdot AFT(j) \cdot LENGTH(j-i)$ where $FORE$, AFT, and $LENGTH$ are the scores of the three classifiers. The hypothesis of Bwi is all intervals i, j where this score exceeds some threshold.

Bwi uses boosting to learn the fore and aft classifiers. Boosting builds a classifier by repeatedly calling a *base learner* with different weightings of the training set. For Bwi, the *base learner* greedily searches for a rule that distinguishes the fore (or aft) tokens from the non-fore (or non-aft) tokens, based on patterns involving the *prefix* (tokens preceding) and the *suffix* (tokens including and following) an example token. For instance, the rule *PREFIX*=⟨"."⟩, *SUFFIX*=⟨"(", OneCharToken, ")"⟩, would match the left parenthesis of all bullet-style image pointers from Figures 1. In Bwi, the primitive conditions in a prefix (or suffix) pattern can test for equality to a specific token (like "." above) or may include certain predefined *wildcard conditions* (like OneCharToken above). Rules are found by repeatedly adding to the prefix (or suffix) pattern the extension of length L or less which maximizes a scoring function. The number L determines the amount of "lookahead" used in the greedy search.

3.3.2 A simple learning method for extraction.

We experimented with a modified version of Bwi, which we will call aBwi.[3] The main difference is that, while Bwi's base learner uses a token-sequence representation of a document, aBwi's base learner uses a feature vector representation.

aBwi is based on two main components. One component is a feature-vector learning system that closely follows the boosting algorithm and base learner in Bwi: this is implemented using Slipper [19], a feature-vector rule learner to which Bwi is closely related.[4] The second is a tool called Peel[5], which constructs a proposition representation of a document from a certain intermediate form.

Specifically, Peel inputs a number of of *labeled substring sets*, and produces a set of labeled feature vectors. A *labeled substring* is a tuple $\langle f, i, j, \ell \rangle$ where f is a file name, i and j are character indices into the file, and ℓ is a label associated with the substring of f between i and j. Empty substrings are allowed. A *labeled substring set* $\mathcal{S}$ is simply a set of labeled substrings.

One labeled substring set is designated as the *example set*, and the others are called *feature sets*. Feature sets are generated in Peel by taking each substring from the example set in turn, and following a special *feature generation program*, each step of which is of the form "$emit(S_i, DIR_i, DIST_i, OP_i)$", where S_i is (the name of) a labeled substring set, DIR_i is a *direction*, $DIST_i$ is a *distance*, and OP_i is an *operation*. Each *emit* function produces to a single feature. To compute this feature, Peel navigates from the example substring x (using the direction and distance parameters) to another labeled substring y, typically a feature substring, and then applies the given operation the pair (x, y). The produces a result v, which is the value (in the feature vector) of the feature corresponding to the emit function.

As an example, suppose that $\mathcal{S}_{token}$ is a labeled substring set containing all tuples $\langle f, i, j, T \rangle$ where T is the token appearing at position i, j in f. (For instance, if f contains the caption of Figure 1, then $\mathcal{S}_{token}$ might contain $\langle f, 1, 3,$ "Fig"$\rangle$, $\langle f, 3, 4,$ "."$\rangle$, $\langle f, 5, 6,$ "1"$\rangle$, $\langle f, 6, 7,$ "."$\rangle$, $\langle f, 8, 14,$ "Kinase"$\rangle, \ldots$. Then the program

emit(token,before,−2,label), *emit(token,before,−1,label)*, *emit(token,inside,0,label)*,
emit(token,after,+1,label), *emit(token,after,+2,label)*,

would emit ⟨'cytosol','.','(','A',')'⟩ for the "fore" token of the first image pointer in Figure 1.

Peel currently supports the directions *before*, *after*, and *inside*, and the operations *label* (which produces the label of the feature substring x) and *distanceBB*, *distanceBE*, *distanceEB*, and *distanceEE* (which produce the distance from the beginning of x to the beginning of y, the beginning of x to the end of y, and so on). However, this small set of operations provides great flexibility when combined with the ability to construct and use arbitrary labeled substring sets. For instance, to generate features corresponding to the OneCharToken wildcard of Bwi, one could construct a new set called *isOneCharToken* by copying $\mathcal{S}_{token}$ and changing each label to "true" or "false", depending on token length.

Using Peel and an appropriate feature-vector learner one can construct a reasonable approximation to Bwi. In aBwi patterns must of a bounded length, which is determined by the user when feature-vectors are constructed; in contrast, Bwi's base learner selects pattern length dynamically during learning. However, aBwi allows more flexibility in feature engineering (as we will discuss below) and also makes it simple to explore different underlying learning methods.

[3]For "Almost BWI."

[4]The learning component is implemented using the Slipper options `-ag -n100 -S0 -i0`.

[5]For "Preparing Examples for Extraction Learning."

aBwi differs from Bwi in three other ways. One difference is a matter of usage: rather than classifing tokens as fore and aft and using an aggregate score, we elected to classify the substrings produced by HandCode2 as image-pointer or non-image pointer.[6] A second difference is that the aBwi base learner does not perform lookahead, but does allow a pattern to contain conditions about any token in the window; in contrast Bwi's base learner does use lookahead, but only produces patterns based on token sequences that are contiguous and starting at the token to be classified. A third difference is that we used a different set of wildcards than Bwi, based on our intuitions of the caption-understanding problem.[7]

aBwi has two parameters, window size (W) and the number of rounds of boosting (T). Bwi also has two parameters, lookahead (L) and rounds of boosting (T).

3.3.3 Experimental results with token-window features

Initial results for extraction. We used 10-fold cross-validation to evaluate the performance of aBwi, initially fixing T=100 and varying the window size W. aBwi outputs a confidence on each prediction, and by varying a threshold one can trade off recall for precision. Setting this threshold to maximize F-measure for W=2 gives a recall of 89.7% and precision of 91.0%, for an F-measure of 90.3%. Using the default threshold (which minimizes error rate) gives essentially identical results.[8]

aBwi seems to be more sensitive to W than T. F-measure varied widely as W was varied from 1 to 10, with performance peaking at W=2 (not all results for W are shown). For W=2, T=100 seems approximately optimal, as shown in the graph accompanying Figure 2.

Additional extraction results. An advantage of aBwi is that is possible to easily modify the system by changing the learning component. As an experiment, we replaced the learner in aBwi with three other learning systems: Slipper (with its default options, including mechanisms to handle noisy data and internal cross-validation to set T); Ripper [20], and SVM Light [21]. Some representative results are shown in Figure 2 under the method name NrBwi.[9] On this problem, none of the other learners improves over aBwi, although all outperform the hand-coded systems.

Classification results. As discussed above, for some purposes it is sufficient to simply identify (*i.e.*, extract) image pointers. For other tasks, however, it is necessary to *classify* extracted image pointers as *bullet-style*, *citation-style*, or *NP-style*.

The most natural approach doing this classification is in two stages: first extraction, and then classification. However, since the extraction is based on an underlying classifier, it is just as easy to perform the task in one step, in which a single multi-label classifier is used to label substrings as bullet-style, citation-style, NP-style, or "other" (where "other" indicates that a substring is not an image pointer at all.) Figure 3 shows results obtained using this combined classification/extraction approach, which has provided the best performance so far. All results are for aBwi with T=100, and the best results were obtained with W=3.

[6] *I.e.*, the HandCode2 substrings formed the example set, rather than using all possible fore/aft pointers as examples. Although the general approach allows learning classifiers for fore/aft tokens as well, it was inappropriate to use this general method here, given the performance of HandCode2.

[7] Specifically our wildcards are OneCharToken, isPunctuation, isAllDigits, isLowerCase, isUpperCase and isInitialCap.

[8] Recall 88.4%, precision 92.0%, and F-measure 90.2%.

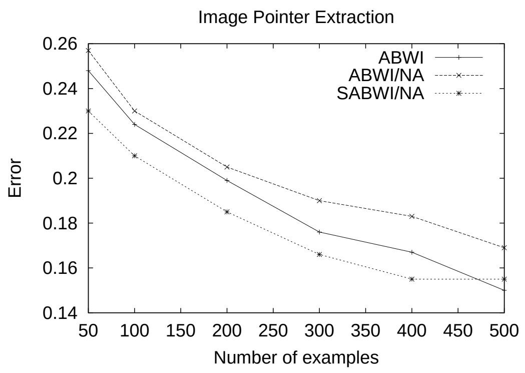
[9] For "Not Really Bwi." Lines tagged with an asterisk are for learners with settings optimized for error rate rather than F-measure.

Method	Error rate		
	W=2	W=3	W=5
aBwi	24.6	27.5	26.7
aBwi/na	26.7	22.2	26.7
saBwi/na	24.2	**18.2**	22.6

Confusion matrix for saBwi/na (W=3)

		Predicted			
		Bullet	Cite	NP	Other
	Bullet-style	191	4	0	0
Actual	Citation-style	6	313	15	10
	NP-style	1	5	16	1
	Other	4	72	18	92

Figure 3: Left: performance of aBwi variants on simultaneous extraction and classification. Right: details of performance of saBwi/na, the best-performing caption-understanding method.



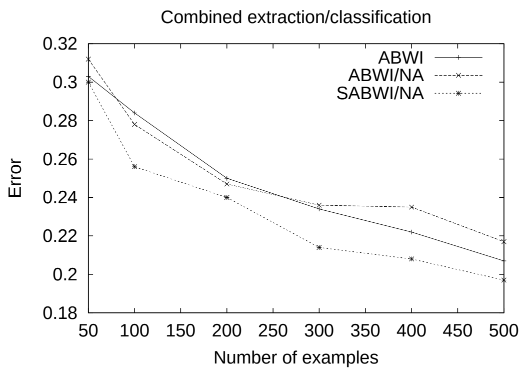


Figure 4: Performance of aBwi, aBwi/na, and saBwi/na on extraction (left) and combined classification and extraction (right) on subsamples of the data.

3.3.4 Experimental results with additional features

An advantage of aBwi is that Peel makes it relatively easy to introduce additional features. To explore the use of engineered features, we added two additional labeled string sets. First, the set $\mathcal{S}_{sentence}$ was populated with empty strings indicating sentence boundaries. Second, the $\mathcal{S}_{special}$ string was created for other "special" substrings that we conjectured would be useful. Specifically, $\mathcal{S}_{special}$ contains a *handCode1* label for any substring matching one of the patterns used in the HandCode1 method; a *color* label for substring corresponding to colors like "red", "green", etc; a *measure* label for common measurements (like "mg", "ml", etc); a *citation* label for all four-digit numbers between 1980 and 2003 and the substring "et al"; and a *place* label for words like "top", "bottom", "left", "right", etc. We then augmented the Bwi-emulating Peel script to output additional features corresponding to the labels of any "special" string found *inside* the candidate image pointer, and the distance to the previous sentence boundary.

The results are shown in the bottom table in Figure 2 (for extraction) and Figure 3 (for classification) under the method name aBwi/na.[10] Performance was not consistently improved, and we conjecture the reason is that while Bwi is designed to learn rules that are positively correlated with the target strings, many of the features described above are negatively correlated. We modified the underlying learning system to include the negation of all tests allowed previously, and also to learn rules that reject non-target strings.[11] This "symmetric" version of aBwi (in the tables, saBwi/na) improves performance consistently over "vanilla" aBwi, and outperforms the best previous systems on both the extraction and classification tasks.

[10]For aBwi with New Attributes.

[11]We used the Slipper options `-au -n100 -S0 -i0 -!ns`.

3.3.5 Additional details on performance

In Figure 4, we show the performance and aBwi, aBwi/na, and saBwi/na on the extraction and classification tasks using subsamples of the data. All curves show error rates averaged over 20 independent trials with W=3 and T=100. For extraction, the additional features are harmfull for aBwi but helpful for saBwi/na, especially with less training data. For classification, additional features show mixed results for aBwi, but lead to improvement with saBwi/na. The curves also indicate that performance on the classification task might be improved by simply collecting more labeled data.

On the right in Figure 3, we show the full confusion matrix of the best classifier (on the cross-validated test examples). Performance is extremely good (recall of 98% and precision of 94.6%) on bullet-style labels, which are the ones most likely to severely impact performance. Most errors are made by incorrectly rejecting citation-style image pointers.

4 Concluding Remarks

In many genres of scientific publication, caption text is extremely dense in information. However, applying automated text extraction methods to caption text can be difficult. We have considered the problem of "understanding" captions, in the sense of extracting and analyzing their extra-grammatical structure of captions. Specifically, we proposed extracting *image pointer* text and classifying it into three categories, *bullet-style*, *citation-style*, and *NP-style*.

In other contexts, extra-grammatical structure has proven useful for extraction and classification [22, 23]. In the specific case of biomedical captions, there are a number of reasons for being interested in this sort of "understanding". Based on this analysis, one can recover the original caption text, *sans* extra-grammatical insertions, which might facilitate later text processing. More importantly, one can also associate specific parts of the caption text with specific parts of the image, by using relatively simple scoping rules. This means textual information from captions can be used to leverage automated understanding of the associated image—a goal of our own ongoing work in analysis of fluorescence microscope images appearing in on-line journals [15]. The same sort of text-image associations might also be used for other purposes, such as to facilitate content-based image retrieval (CBIR) of scientific images.

In an experimental study with a hand-labeled corpus of figures, we evaluated a number of extraction and classification techniques. The best-performing method is the novel extraction system saBwi/na, which naturally extends an earlier extraction system Bwi with the ability to use features based on arbitrary labeled substrings. Exploiting this ability we included a number of substring features engineered specifically for the caption-understanding task. The lead to improvements in performance over hand-coded extraction methods and competitive "off-the-shelf" learning methods. The best system described is usefully accurate, and obtains both recall and precision in excess of 94% on bullet-style image pointers, the most important class in the combined extraction/classification task.

References

[1] Blaschke, C., Andrade, M. A., Ouzounis, C., and Valencia, A.: Automatic extraction of biological information from scientific text: Protein-protein interactions. In *Proceedings of the 1999 International Conference on Intelligent Systems for Molecular Biology (ISMB-99).* 1999, pp 60–67.

[2] Sekimizu, T., Park, H., and Tsujii, J. Identifying the interaction between genes and gene products based on frequently seen verbs in Medline abstracts. In *Genome Informatics*, pp 62–71. Universal Academy Press, Inc, 1998.

[3] Pustejovsky, J., Castaño, J., Zhang, J., Kotecki, M., and Cochran, B.: Robust relational parsing over biomedical literature: Extracting inhibit relations. In *Proceedings of 2002 the Pacific Symposium on Biocomputing (PSB-2002).* 2002, pp 362–373.

[4] Thomas, J., Milward, D., Ouzounis, C., Pulman, S., and Carroll, M.: Automatic extraction of protein interactions from scientific abstracts. In *Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000).* 2000, pp 538–549.

[5] Stephens, M., Palakal, M., Mukhopadhyay, S., Raje, R., and Mostafa, J.: Detecting gene relations from medline abstracts. In *Pacific Symposium on Biocomputing.* 2001, pp 483–496.

[6] Humphreys, K., Demetriou, G., and Gaizauskas, R.: Two applications of information extraction to biological science journal articles: Enzyme interactions and protein structures. In *Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000).* 2000, pp 502–513.

[7] Fukuda, K., Tsunoda, T., Tamura, A., and Takagi, T.: Toward information extraction: Identifying protein names from biological papers. In *Proceedings of 1998 the Pacific Symposium on Biocomputing (PSB-1998).* 1998, pp 707–718.

[8] Rindflesch, T., Tanabe, L., Weinstein, J. N., and Hunter, L.: Edgar: Extraction of drugs, genes and relations from the biomedical literature. In *Proceedings of 2000 the Pacific Symposium on Biocomputing (PSB-2000).* 2000, pp 514–525.

[9] Bunescu, R., Ge, R., Mooney, R. J., Marcotte, E., and Ramani, A. K. Extracting gene and protein names from biomedical abstracts. Unpublished Technical Note, Available from http://www.cs.utexas.edu/users/ml/publication/ie.html, 2002.

[10] Craven, M. and Kumlien, J.: Constructing biological knowledge bases by extracting information from text sources. In *Proceedings of the 7th International Conference on Intelligent Systems for Molecular Biology (ISMB-99).* AAAI Press, 1999, pp 77–86.

[11] Stapley, B., Kelley, L., and Sternberg, M.: Predicting the sub-cellular location of proteins from text using support vector machines. In *Proceedings of the 2002 Pacific Symposium on Biocomputing.* 2002, pp 374–385.

[12] Boland, M. V. and Murphy, R. F.: A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of hela cells. *Bioinformatics.* 17: 1213–1223, 2001.

[13] Velliste, M. and Murphy, R. F.: Automated determination of protein subcellular locations from 3d fluorescence microscope images. In *Proceedings of the 2002 IEEE International Symposium on Biomedical Imaging (ISBI-2002).* 2002, pp 867–870.

[14] Murphy, R. F., Velliste, M., and Porreca, G.: Robust classification of subcellular location patterns in fluorescence microscope images. In *Proceedings of the 2002 IEEE International Workshop on Neural Networks for Signal Processing.* 2002, pp 67–76.

[15] Murphy, R. F., Velliste, M., and Porreca, G.: Searching online journals for fluorescence microscope images depicting protein subcellular location patterns. In *Proceedings of the 2nd IEEE International Symposium on Bio-informatics and Biomedical Engineering (BIBE-2001).* 2001, pp 119–128.

[16] R., A. A., C., R. T., and C., B. A.: Exploiting a large thesaurus for information retrieval. In *Proceedings of RIAO 94.* 1994, pp 197–216.

[17] Daelemans, W., Buchholz, S., and Veenstra, J.: Memory-based shallow parsing. In *Proceedings of the EACL'99 workshop on Computational Natural Language Learning (CoNLL-99).* Bergen, Norway, 1999.

[18] Freitag, D. and Kushmeric, N.: Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000).* Austin, TX, 2000.

[19] Cohen, W. W. and Singer, Y.: A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99).* Orlando, FL, 1999.

[20] Cohen, W. W.: Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference.* Lake Tahoe, California, Morgan Kaufmann, 1995.

[21] Joachims, T.: *Learning to Classify Text Using Support Vector Machines.* Kluwer, 2002.

[22] Blei, D. M., Bagnell, J. A., and McCallum, A. K.: Learning with scope, with application to information extraction and classification. In *Proceedings of UAI-2002.* Edmonton, Alberta, 2002.

[23] Cohen, W. W.: Improving a page classifier with anchor extraction and link analysis. In *Advances in Neural Processing Systems 15.* Vancouver, British Columbia, 2002.