Empirically Estimating Order Constraints for Content Planning in Generation

Pablo A. Duboue and **Kathleen R. McKeown**

Computer Science Department
Columbia University
10027, New York, NY, USA
{pablo,kathy}@cs.columbia.edu

Abstract

In a language generation system, a content planner embodies one or more "plans" that are usually hand–crafted, sometimes through manual analysis of target text. In this paper, we present a system that we developed to automatically learn elements of a plan and the ordering constraints among them. As training data, we use semantically annotated transcripts of domain experts performing the task our system is designed to mimic. Given the large degree of variation in the spoken language of the transcripts, we developed a novel algorithm to find parallels between transcripts based on techniques used in computational genomics. Our proposed methodology was evaluated two–fold: the learning and generalization capabilities were quantitatively evaluated using cross validation obtaining a level of accuracy of 89%. A qualitative evaluation is also provided.

1 Introduction

In a language generation system, a content planner typically uses one or more "plans" to represent the content to be included in the output and the ordering between content elements. Some researchers rely on generic planners (e.g., (Dale, 1988)) for this task, while others use plans based on Rhetorical Structure Theory (RST) (e.g., (Bouayad-Aga et al., 2000; Moore and Paris, 1993; Hovy, 1993)) or schemas (e.g., (McKeown, 1985; McKeown et al., 1997)). In all cases, constraints on application of rules (e.g., plan operators), which determine content and order, are usually hand-crafted, sometimes through manual analysis of target text.

In this paper, we present a method for learning the basic patterns contained within a plan and the ordering among them. As training data, we use semantically tagged transcripts of domain experts performing the task our system is designed to mimic, an oral briefing of patient status after undergoing coronary bypass surgery. Given that our target output is spoken language, there is some level of variability between individual transcripts. It is difficult for a human to see patterns in the data and thus supervised learning based on hand-tagged training sets can not be applied. We need a learning algorithm that can discover ordering patterns in apparently unordered input.

We based our unsupervised learning algorithm on techniques used in computational genomics (Durbin et al., 1998), where from large amounts of seemingly unorganized genetic sequences, patterns representing meaningful biological features are discovered. In our application, a transcript is the equivalent of a sequence and we are searching for patterns that occur repeatedly across multiple sequences. We can think of these patterns as the basic elements of a plan, representing small clusters of semantic units that are similar in size, for example, to the nucleus-satellite pairs of RST.[1] By learning ordering constraints over these ele-

[1]Note, however, that we do not learn or represent intention.

age, gender, pmh, pmh, pmh, pmh, med-preop, med-preop, med-preop, drip-preop, med-preop, ekg-preop, echo-preop, hct-preop, procedure, . . .

Figure 2: The semantic sequence obtained from the transcript shown in Figure 1.

ments, we produce a plan that can be expressed as a constraint-satisfaction problem. In this paper, we focus on learning the plan elements and the ordering constraints between them. Our system uses combinatorial pattern matching (Rigoutsos and Floratos, 1998) combined with clustering to learn plan elements. Subsequently, it applies counting procedures to learn ordering constraints among these elements.

Our system produced a set of 24 schemata units, that we call "plan elements"[2], and 29 ordering constraints between these basic plan elements, which we compared to the elements contained in the orginal hand-crafted plan that was constructed based on hand-analysis of transcripts, input from domain experts, and experimental evaluation of the system (McKeown et al., 2000).

The remainder of this article is organized as follows: first the data used in our experiments is presented and its overall structure and acquisition methodology are analyzed. In Section 3 our techniques are described, together with their grounding in computational genomics. The quantitative and qualitative evaluation are discussed in Section 4. Related work is presented in Section 5. Conclusions and future work are discussed in Section 6.

2 Our data

Our research is part of MAGIC (Dalal et al., 1996; McKeown et al., 2000), a system that is designed to produce a briefing of patient status after undergoing a coronary bypass operation. Currently, when a patient is brought to the intensive care unit (ICU) after surgery, one of the residents who was present in the operating room gives a briefing to the ICU nurses and residents. Several of these briefings were collected and annotated for the aforementioned evaluation. The resident was equipped with a wearable tape recorder to tape the briefings, which were transcribed to provide the base of our empirical data. The text was subsequently annotated with semantic tags as shown in Figure 1. The figure shows that each sentence is split into several semantically tagged chunks. The tag-set was developed with the assistance of a domain expert in order to capture the different information types that are important for communication and the tagging process was done by two non-experts, after measuring acceptable agreement levels with the domain expert (see (McKeown et al., 2000)). The tag-set totalled over 200 tags. These 200 tags were then mapped to 29 categories, which was also done by a domain expert. These categories are the ones used for our current research.

From these transcripts, we derive the sequences of semantic tags for each transcript. These sequences constitute the input and working material of our analysis, they are an average length of 33 tags per transcript ($\min = 13$, $\max = 66$, $\sigma = 11.6$). A tag-set distribution analysis showed that some of the categories dominate the tag counts. Furthermore, some tags occur fairly regularly towards either the beginning (e.g., **date-of-birth**) or the end (e.g., **urine-output**) of the transcript, while others (e.g., **intraop-problems**) are spread more or less evenly throughout.

Getting these transcripts is a highly expensive task involving the cooperation and time of nurses and physicians in the busy ICU. Our corpus contains a total number of 24 transcripts. Therefore, it is important that we develop techniques that can detect patterns without requiring large amounts of data.

3 Methods

During the preliminary analysis for this research, we looked for techniques to deal with analysis of regularities in sequences of finite items (semantic tags, in this case). We were interested in developing techniques that could scale as well as work with small amounts of highly varied sequences.

Computational biology is another branch of computer science that has this problem as one topic of study. We focused on **motif** detection techniques as a way to reduce the complexity of the overall setting of the problem. In biological

[2]These units can be loosely related to the concept of *messages* in (Reiter and Dale, 2000).

He is $\underbrace{\text{58-year-old}}_{\text{age}}$ $\underbrace{\text{male}}_{\text{gender}}$. History is significant for $\underbrace{\text{Hodgkin's disease}}_{\text{pmh}}$, treated with ...to his neck, back and chest. $\underbrace{\text{Hyperspadias}}_{\text{pmh}}$, $\underbrace{\text{BPH}}_{\text{pmh}}$, $\underbrace{\text{hiatal hernia}}_{\text{pmh}}$ and $\underbrace{\text{proliferative lymph edema in his right arm}}_{\text{pmh}}$. No IV's or blood pressure down in the left arm. Medications — $\underbrace{\text{Inderal}}_{\text{med-preop}}$, $\underbrace{\text{Lopid}}_{\text{med-preop}}$, $\underbrace{\text{Pepcid}}_{\text{med-preop}}$, $\underbrace{\text{nitroglycerine}}_{\text{drip-preop}}$ and $\underbrace{\text{heparin}}_{\text{med-preop}}$. $\underbrace{\text{EKG has PAC's}}_{\text{ekg-preop}}$. $\underbrace{\text{His Echo showed AI, MR of 47 cine amps with hypokinetic basal and anterior apical region.}}_{\text{echo-preop}}$ $\underbrace{\text{Hematocrit 1.2}}_{\text{hct-preop}}$, otherwise his labs are unremarkable. Went to OR for what was felt to be $\underbrace{\text{2 vessel CABG off pump both mammaries}}_{\text{procedure}}$ ......

Figure 1: An annotated transcription of an ICU briefing (after anonymising).

terms, a motif is a small subsequence, highly conserved through evolution. From the computer science standpoint, a motif is a fixed-order pattern, simply because it is a subsequence. The problem of detecting such motifs in large databases has attracted considerable interest in the last decade (see (Hudak and McClure, 1999) for a recent survey). Combinatorial pattern discovery, one technique developed for this problem, promised to be a good fit for our task because it can be parameterized to operate successfully without large amounts of data and it will be able to identify domain swapped motifs: for example, given a–b–c in one sequence and c–b–a in another. This difference is central to our current research, given that order constraints are our main focus. TEIRESIAS (Rigoutsos and Floratos, 1998) and SPLASH (Califano, 1999) are good representatives of this kind of algorithm. We used an adaptation of TEIRESIAS.

The algorithm can be sketched as follows: we apply combinatorial pattern discovery (see Section 3.1) to the semantic sequences. The obtained patterns are refined through clustering (Section 3.2). Counting procedures are then used to estimate order constraints between those clusters (Section 3.3).

3.1 Pattern detection

In this section, we provide a brief explanation of our pattern discovery methodology. The explanation builds on the definitions below:

$\langle L, W\rangle$ pattern. Given that Σ represents the semantic tags alphabet, a pattern is a string of the form $\Sigma\,(\Sigma|?)^*\,\Sigma$, where $?$ represents a *don't care* (wildcard) position. The $\langle L, W\rangle$ parameters are used to further control the amount and placement of the *don't cares*: every subsequence of length W, at least L positions must be filled (i.e., they are non-wildcards characters). This definition entails that $L \leq W$ and also that a $\langle L, W\rangle$ pattern is also a $\langle L, W+1\rangle$ pattern, etc.

Support. The support of pattern p given a set of sequences S is the number of sequences that contain at least one match of p. It indicates how useful a pattern is in a certain environment.

Offset list. The offset list records the matching locations of a pattern p in a list of sequences. They are sets of ordered pairs, where the first position records the sequence number and the second position records the offset in that sequence where p matches (see Figure 3).

Specificity. We define a partial order relation on the pattern space as follows: a pattern p is said to be more specific than a pattern q if: (1) p is equal to q in the defined positions of q but has fewer undefined (i.e., wildcards) positions; or (2) q is a substring of p. Specificity provides a notion of complexity of a pattern (more specific patterns are more complex). See Figure 4 for an example.

Using the previous definitions, the algorithm reduces to the problem of, given a set of sequences, L, W, a minimum windowsize, and a *support*

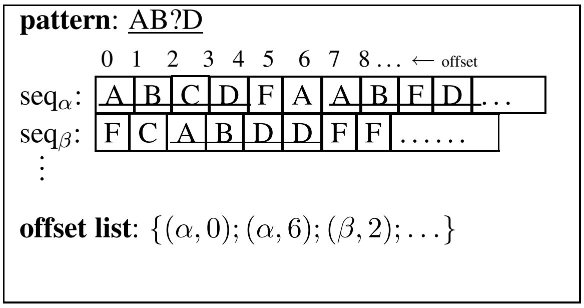

Figure 3: A pattern, a set of sequences and an offset list.

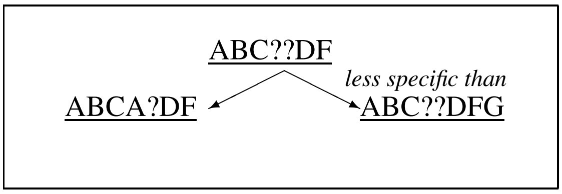

Figure 4: The specificity relation among patterns.

threshold, finding maximal $\langle L, W \rangle$-patterns with at least a support of *support threshold*. Our implementation can be sketched as follows:

Scanning. For a given window size n, all the possible subsequences (i.e., *n*-grams) occurring in the training set are identified. This process is repeated for different window sizes.

Generalizing. For each of the identified subsequences, patterns are created by replacing valid positions (i.e., any place but the first and last positions) with wildcards. Only $\langle L, W \rangle$ patterns with support greater than *support threshold* are kept. Figure 5 shows an example.

Filtering. The above process is repeated increasing the window size until no patterns with enough support are found. The list of identified patterns is then filtered according to specificity: given two patterns in the list, one of them more specific than the other, if both have offset lists of equal size, the less specific one is pruned[3]. This gives us the list of **maximal** motifs (i.e. patterns) which are supported by the training data.

[3]Since they match in exactly the same positions, we prune the less specific one, as it adds no new information.

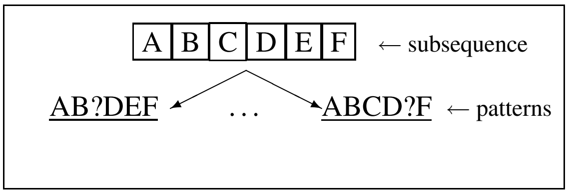

Figure 5: The process of generalizing an existing subsequence.

3.2 Clustering

After the detection of patterns is finished, the number of patterns is relatively large. Moreover, as they have fixed length, they tend to be pretty similar. In fact, many tend to have their support from the same subsequences in the corpus. We are interested in syntactic similarity as well as similarity in context.

A convenient solution was to further cluster the patterns, according to an **approximate matching** distance measure between patterns, defined in an appendix at the end of the paper.

We use agglomerative clustering with the distance between clusters defined as the maximum pairwise distance between elements of the two clusters. Clustering stops when no inter-cluster distance falls below a user-defined threshold.

Each of the resulting clusters has a single pattern represented by the centroid of the cluster. This concept is useful for visualization of the cluster in qualitative evaluation.

3.3 Constraints inference

The last step of our algorithm measures the frequencies of all possible order constraints among pairs of clusters, retaining those that occur often enough to be considered important, according to some relevancy measure. We also discard any constraint that it is violated in any training sequence. We do this in order to obtain clear-cut constraints. Using the number of times a given constraint is violated as a quality measure is a straight-forward extension of our framework. The algorithm proceeds as follows: we build a table of counts that is updated every time a pair of patterns belonging to particular clusters are matched. To obtain clear-cut constraints, we do not count overlapping occurrences of patterns.

From the table of counts we need some relevancy measure, as the distribution of the tags is skewed. We use a simple heuristic to estimate a relevancy measure over the constraints that are never contradicted. We are trying to obtain an estimate of

$$Pr\,(A \prec_{\text{precedes}} B)$$

from the counts of

$$c = A \,\tilde{\precsim}_{\text{preceded}}\, B$$

We normalize with these counts (where x ranges over all the patterns that match before/after A or B):

$$c_1 = A \,\tilde{\precsim}_{\text{preceded}}\, x$$

and

$$c_2 = x \,\tilde{\precsim}_{\text{preceded}}\, B$$

The obtained estimates, $e_1 = c/c_1$ and $e_2 = c/c_2$, will in general yield different numbers. We use the arithmetic mean between both, $e = \frac{(e_1+e_2)}{2}$, as the final estimate for each constraint. It turns out to be a good estimate, that predicts accuracy of the generated constraints (see Section 4).

4 Results

We use cross validation to quantitatively evaluate our results and a comparison against the plan of our existing system for qualitative evaluation.

4.1 Quantitative evaluation

We evaluated two items: how effective the patterns and constraints learned were in an unseen test set and how accurate the predicted constraints were. More precisely:

Pattern Confidence. This figure measures the percentage of identified patterns that were able to match a sequence in the test set.

Constraint Confidence. An ordering constraint between two clusters can only be checkable on a given sequence if at least one pattern from each cluster is present. We measure the percentage of the learned constraints that are indeed checkable over the set of test sequences.

Constraint Accuracy. This is, from our perspective, the most important judgement. It measures the percentage of checkable ordering constraints that are correct, i.e., the order constraint was maintained in **any** pair of matching patterns from both clusters in **all** the test-set sequences.

Table 1: Evaluation results.

Test	Result
pattern confidence	84.62%
constraint confidence	66.70%
constraint accuracy	89.45%

Using 3-fold cross-validation for computing these metrics, we obtained the results shown in Table 1 (averaged over 100 executions of the experiment). The different parameter settings were defined as follows: for the motif detection algorithm $\langle L, W\rangle = \langle 2, 3\rangle$ and *support threshold* of 3. The algorithm will normally find around 100 maximal motifs. The clustering algorithm used a relative distance threshold of 3.5 that translates to an actual treshold of 120 for an average inter-cluster distance of 174. The number of produced clusters was in the order of the 25 clusters or so. Finally, a threshold in relevancy of 0.1 was used in the constraint learning procedure. Given the amount of data available for these experiments all these parameters were hand-tunned.

4.2 Qualitative evaluation

The system was executed using all the available information, with the same parametric settings used in the quantitative evaluation, yielding a set of 29 constraints, out of 23 generated clusters.

These constraints were analyzed by hand and compared to the existing content-planner. We found that most rules that were learned were validated by our existing plan. Moreover, we gained placement constraints for two pieces of semantic information that are currently not represented in the system's plan. In addition, we found minor order variation in relative placement of two different pairs of semantic tags. This leads us to believe that the fixed order on these particular tags can be relaxed to attain greater degrees of variability in the generated plans. The process of creation of the existing content-planner was thorough, informed by multiple domain experts over a three year period. The fact that the obtained constraints mostly occur in the existing plan is very encouraging.

5 Related work

As explained in (Hudak and McClure, 1999), motif detection is usually targeted with alignment techniques (as in (Durbin et al., 1998)) or with combinatorial pattern discovery techniques such as the ones we used here. Combinatorial pattern discovery is more appropriate for our task because it allows for matching across patterns with permutations, for representation of wild cards and for use on smaller data sets.

Similar techniques are used in NLP. Alignments are widely used in MT, for example (Melamed, 1997), but the crossing problem is a phenomenon that occurs repeatedly and at many levels in our task and thus, this is not a suitable approach for us.

Pattern discovery techniques are often used for information extraction (e.g., (Riloff, 1993; Fisher et al., 1995)), but most work uses data that contains patterns labelled with the semantic slot the pattern fills. Given the difficulty for humans in finding patterns systematically in our data, we needed unsupervised techniques such as those developed in computational genomics.

Other stochastic approaches to NLG normally focus on the problem of sentence generation, including syntactic and lexical realization (e.g., (Langkilde and Knight, 1998; Bangalore and Rambow, 2000; Knight and Hatzivassiloglou, 1995)). Concurrent work analyzing constraints on ordering of sentences in summarization found that a coherence constraint that ensures that blocks of sentences on the same topic tend to occur together (Barzilay et al., 2001). This results in a bottom-up approach for ordering that opportunistically groups sentences together based on content features. In contrast, our work attempts to automatically learn plans for generation based on semantic types of the input clause, resulting in a top-down planner for selecting and ordering content.

6 Conclusions

In this paper we presented a technique for extracting order constraints among plan elements that performs satisfactorily without the need of large corpora. Using a conservative set of parameters, we were able to reconstruct a good portion of a carefully hand-crafted planner. Moreover, as discussed in the evaluation, there are several pieces of information in the transcripts which are not present in the current system. From our learned results, we have inferred placement constraints of the new information in relation to the previous plan elements without further interviews with experts.

Furthermore, it seems we have captured order-sensitive information in the patterns and free-order information is kept in the *don't care* model. The patterns, and ordering constraints among them, provide a backbone of relatively fixed structure, while *don't care*s are interspersed among them. This model, being probabilistic in nature, means a great deal of variation, but our generated plans should have variability in the right positions. This is similar to findings of *floating positioning* of information, together with *oportunistic rendering* of the data as used in STREAK (Robin and McKeown, 1996).

6.1 Future work

We are planning to use these techniques to revise our current content-planner and incorporate information that is learned from the transcripts to increase the possible variation in system output.

The final step in producing a full-fledged content-planner is to add semantic constraints on the selection of possible orderings. This can be generated through clustering of semantic input to the generator.

We also are interested in further evaluating the technique in an unrestricted domain such as the Wall Street Journal (WSJ) with shallow semantics such as the WordNet top-category for each NP-head. This kind of experiment may show strengths and limitations of the algorithm in large corpora.

7 Acknowledgments

This research is supported in part by NLM Contract R01 LM06593-01 and the Columbia University Center for Advanced Technology in Information Management (funded by the New York State Science and Technology Foundation). The authors would like to thank Regina Barzilay,

intraop-problems	**intraop-problems**	**operation** 11.11% **drip** 33.33% **intraop-problems** 33.33% **total-meds-anesthetics** 22.22%	**drip**
intraop-problems	**operation** 14.29% **drip** 14.29% **intraop-problems** 42.86% **total-meds-anesthetics** 28.58%	**drip**	**drip**
intraop-problems	**intraop-problems**	**operation** 20.00% **drip** 20.00% **intraop-problems** 20.00% **total-meds-anesthetics** 40.00%	**drip** **drip**

Figure 6: Cluster and patterns example. Each line corresponds to a different pattern. The elements between braces are don't care positions (three patterns conform this cluster: **intraop-problems intraop-problems ? drip**, **intraop-problems ? drip drip** and **intraop-problems intraop-problems drip drip** the don't care model shown in each brace must sum up to 1 but there is a strong overlap between patterns —the main reason for clustering)

Noemie Elhadad and Smaranda Muresan for helpful suggestions and comments. The aid of two anonymous reviewers was also highly appreciated.

References

Srinivas Bangalore and Owen Rambow. 2000. Exploiting a probabilistic hierarchical model for generation. In *COLING, 2000*, Saarbrcken, Germany.

Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. 2001. Sentence ordering in multidocument summarization. In *HLT, 2001*, San Diego, CA.

Nadjet Bouayad-Aga, Richard Power, and Donia Scott. 2000. Can text structure be incompatible with rhetorical structure? In *Proceedings of the 1st International Conference on Natural Language Generation (INLG-2000)*, pages 194–200, Mitzpe Ramon, Israel.

Andrea Califano. 1999. Splash: Structural pattern localization analysis by sequential histograms. *Bioinformatics*, 12, February.

Mukesh Dalal, Steven Feiner, , Kathleen McKeown, ShiMei Pan, Michelle Zhou, Tobias Hollerer, James Shaw, Yong Feng, and Jeanne Fromer. 1996. Negotiation for automated generation of temporal multimedia presentations. In *Proceedings of ACM Multimedia '96*, Philadelphia.

Robert Dale. 1988. *Generating referring expressions in a domain of objects and processes*. Ph.D. thesis, University of Edinburgh.

Richard Durbin, S. Eddy, A. Krogh, and G. Mitchison. 1998. *Biological sequence analysis*. Cambridge Univeristy Press.

David Fisher, Stephen Soderland, Joseph McCarthy, Fangfang Feng, and Wendy Lehnert. 1995. Description of the umass system as used for muc-6. In Morgan Kaufman, editor, *Proceedings of the Sixth Message Understanding Conference (MUC-6)*, pages 127–140, San Francisco.

Eduard H. Hovy. 1993. Automated discourse generation using discourse structure relations. *Artificial Intelligence*. (Special Issue on Natural Language Processing).

J. Hudak and Marcela McClure. 1999. A comparative analysis of computational motif–detection methods. In R.B. Altman, A. K. Dunker, L. Hunter, T. E. Klein, and K. Lauderdale, editors, *Pacific Symposium on Biocomputing, '99*, pages 138–149, New Jersey. World Scientific.

Kevin Knight and Vasileios Hatzivassiloglou. 1995. Two-level, many-paths generation. In *Proceedings of the Conference of the Association for Computational Linguistics (ACL'95)*.

Irene Langkilde and Kevin Knight. 1998. The practical value of n-grams in generation. In *Proceedings of the Ninth International Natural Language Generation Workshop (INLG'98)*.

Kathleen McKeown, ShiMei Pan, James Shaw, Jordan Desmand, and Barry Allen. 1997. Language generation for multimedia healthcare briefings. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP'97)*, Washington, DC, April.

Kathleen R. McKeown, Desmond Jordan, Steven Feiner, James Shaw, Elizabeth Chen, Shabina Ahmad, Andre Kushniruk, and Vimla Patel. 2000. A study of communication in the cardiac surgery intensive care unit and its implications for automated briefing. In *AMIA '2000*.

Kathleen R. McKeown. 1985. *Text Generation: Using Discourse Strategies and Focus Constraints to Generate Natural Language Text.* Cambridge University Press.

I. Dan Melamed. 1997. A portable algorithm for mapping bitext correspondence. In *35th Conference of the Association for Computational Linguistics (ACL'97)*, Madrid, Spain.

Johanna D. Moore and Cécile L. Paris. 1993. Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics*, 19(4):651–695.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Isidore Rigoutsos and Aris Floratos. 1998. Combinatorial pattern discovery in biological sequences: the teiresias algorithm. *Bioinformatics*, 14(1):55–67.

Ellen Riloff. 1993. Automatically constructing a dictionary for information extraction. In AAAI Press / MIT Press, editor, *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 811–816.

Jacques Robin and Kathleen McKeown. 1996. Empirically designing and evaluating a new revision–based model for summary generation. *Artificial Intelligence*, 85(1–2):135–179.

Appendix - Definition of the distance measure used for clustering.

An approximate matching measure is defined for a given extended pattern. The extended pattern is represented as a sequence of sets; defined positions have a singleton set, while wildcard positions contain the non-zero probability elements in their *don't care* model (e.g. given **intraop-problems, intraop-problems, {drip** 10%,**intubation** 90%}, **drip** we model this as [{**intraop-problems**}; {**intraop-problems**}; {**drip, intubation**}; {**drip**}]).

Consider p to be such a pattern, o an offset and S a sequence, the approximate matching is defined by

$$\hat{m}(p, o, S) = \frac{\sum_{i=0}^{\mathrm{length}(p)} \mathrm{match}(p[i], S[i+o])}{\mathrm{length}(p)}$$

where the $\mathrm{match}(P, e)$ function is defined as 0 if $e \in P$, 1 otherwise, and where P is the set at position i in the extended pattern p and e is the element of the sequence S at position $i + o$.

Our measure is normalized to $[0, 1]$. Using this function, we define the approximate matching distance measure (one way) between a pattern p_1 and a pattern p_2 as the sum (averaged over the length of the offset list of p_1) of all the approximate matching measures of p_2 over the offset list of p_1. This is, again, a real number in $[0, 1]$. To ensure symmetry, we define the distance between p_1 and p_2 as the average between the one way distance between p_1 and p_2 and between p_2 and p_1.