Abstract: Document content (16) is received for rendering in accordance with one or more variable document templates (18). Scores are assigned to respective allocations of the document content (16) to a page. Based on the scores, an optimal number of pages to layout the document content (16) is determined. The following information is determined from the optimal number of pages: a respective optimal allocation of the document content (16) to each of the optimal number of pages, a respective optimal one of the variable document templates for each of the optimal number of pages, and optimal values of parameters defining the relative sizes and positions of the template elements of the ascertained optimal variable document template for each of the optimal number of pages. A document (12) is generated based on the ascertained information.
Electronic documents are created in many different ways. For example, desktop application programs, such as Microsoft Word, Quark Design, and Adobe InDesign, are frequently used to create electronic documents. These electronic documents contain various types of content arranged with a particular layout and style.

Template-based electronic document formats describe a predefined layout arrangement of fields that are capable of accepting variable content. In some approaches, the size, shape and placement of the template fields are fixed. In another approach, a document is represented as an adaptive layout template that contains predefined content areas whose positions and sizes may be varied within specified ranges. In particular, the content areas are defined by variables with respective value domains that define the size, position, and content of the content areas. A user specifies constraints that limit variable ranges and define relationships between variables and values. A constraint solver typically generates a final document that satisfies all of the specified constraints.

Systems and methods for generating documents from variable document templates would be beneficial.

DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of an example of a document processing system for generating documents with variable content.

FIG. 2 is a block diagram of an example of a set of variable document templates.

FIG. 3 is a diagrammatic view of an example of an hierarchical arrangement of templates into collections of a template library.

FIG. 4 shows an example of a process of generating a document layout from document content.

FIG. 5 is a flow diagram of an example of a method of generating a document having an optimal number of pages to which document content is allocated.

FIG. 6 is a diagrammatic view of an example of a representation of a document as a sequence of templates, template parameters, and content allocations that is sampled from an overall probability distribution over P pages.

VARIABLE TEMPLATE BASED DOCUMENT GENERATION

BACKGROUND

Electronic documents are created in many different ways. For example, desktop application programs, such as Microsoft Word, Quark Design, and Adobe InDesign, are frequently used to create electronic documents. These electronic documents contain various types of content arranged with a particular layout and style. Template-based electronic document formats describe a predefined layout arrangement of fields that are capable of accepting variable content. In some approaches, the size, shape and placement of the template fields are fixed. In another approach, a document is represented as an adaptive layout template that contains predefined content areas whose positions and sizes may be varied within specified ranges. In particular, the content areas are defined by variables with respective value domains that define the size, position, and content of the content areas. A user specifies constraints that limit variable ranges and define relationships between variables and values. A constraint solver typically generates a final document that satisfies all of the specified constraints.

Systems and methods for generating documents from variable document templates would be beneficial.

DESCRIPTION OF DRAWINGS

FIG. 1 is a block diagram of an example of a document processing system for generating documents with variable content.

FIG. 2 is a block diagram of an example of a set of variable document templates.

FIG. 3 is a diagrammatic view of an example of an hierarchical arrangement of templates into collections of a template library.

FIG. 4 shows an example of a process of generating a document layout from document content.

FIG. 5 is a flow diagram of an example of a method of generating a document having an optimal number of pages to which document content is allocated.

FIG. 6 is a diagrammatic view of an example of a representation of a document as a sequence of templates, template parameters, and content allocations that is sampled from an overall probability distribution over P pages.
FIG. 7 is a flow diagram of an example of a method of determining an optimal number of pages for allocating document content. FIG. 8 is a block diagram of an example of a computer system.

DETAILED DESCRIPTION

In the following description, like reference numbers are used to identify like elements. Furthermore, the drawings are intended to illustrate major features of like elements. Furthermore, the drawings are intended to illustrate major features of exemplary embodiments in a diagrammatic manner. The drawings are not intended to depict every feature of actual embodiments nor relative dimensions of the depicted elements, and are not drawn to scale.

As used herein a “computer” is any machine, device, or apparatus that processes data according to computer-readable instructions that are stored on a computer-readable medium either temporarily or permanently. A “computer-operating system” is a component of a computer system that manages and coordinates the performance of tasks and the sharing of computing and hardware resources. A “software application” (also referred to as machine readable instructions, software, an application, a computer program, a program, and a computer application) is a set of instructions that a computer can interpret and execute to perform one or more specific tasks. A “data file” is a block of information that durably stores data for use by a software application.

The term “computer-readable medium” refers to any tangible, non-transitory medium capable of storing information that is readable by a machine (e.g., a computer). Storage devices suitable for tangibly embodying these instructions and data include, but are not limited to, all forms of physical, non-transitory computer-readable include, but are not limited to, all forms of physical, non-transitory computer-readable memory, including, for example, semiconductor memory devices, such as random access memory (RAM), EPROM, EEPROM and Flash memory devices, magnetic disks such as internal hard disks and removable hard disks, magneto-optical disks, DVD-ROM/RAM, and CD-ROM/RAM.

A “variable document template” is a data structure that specifies one or more template elements, including at least one template element that is designated for receiving variable document content. A “template element” is any component of a variable document template that is taken into account in an optimization process for determining a final document specification for document content to be inserted into the variable document template. Examples of template elements include content blocks of variable document template.

Examples of template elements include content blocks.
The adaptability of a template element to different document content typically is controlled by at least one adaptive layout parameter that is characterized by one or more respective values that typically are set by a document template designer.

As used herein, the term "page" refers to any type of discrete area in which graphic objects may be laid out, including a physical page embodied by a discrete physical medium (e.g., a piece of paper) on which a layout of graphic objects may be printed, and a virtual, digital or electronic page containing a layout or graphic objects printed, and a virtual, digital or electronic page containing a layout of graphic objects that may be presented to a user by, for example, an electronic display device.

A "window" is a visual area of a display that typically includes a user interface. A window typically displays the output of a machine readable instructions process and typically enables a user to input commands or data for the machine readable instructions process. A window that has a parent is called a "child window." A window that has no parent, or whose parent is the desktop window, is called a "top-level window." A "desktop" is a system-defined window that paints the background of a graphical user interface (GUI) and serves as the base for all windows displayed by all machine readable instructions processes.

Assumed herein, the term "includes" means includes but not limited to, and the term "including" means including but not limited to. The term "based on" means based at least in part on.

The examples that are described herein provide systems and methods for generating documents from variable document templates. These examples are able to generate documents from variable document templates. These examples are able to determine an optimal allocation of document content to an optical number of pages in to determine an optimal allocation of document content to an optical number of pages in a fast and efficient manner. Some of these examples incrementally determine an fast and efficient manner. Some of these examples incrementally determine an optimal allocation of document content over the best previous allocation of a subset of optimal allocation of document content over the best previous allocation of a subset of the document content to any page count. These examples do not require an implicit loop over page count, instead, for each document content increment an optimal page count is updated when the optimal previous subset is found. Due to their efficient use of processing and memory resources, these examples readily may be implemented in a wide variety of application environments, including embedded environments, which are subject to significant processing and memory constraints.

The figures show examples of a document processing system 10 for generating a document 2 with variable content. The document processing system 10 for generating a document 12 with variable content.
The document processing system 10 includes a layout engine 14 that processes electronic document content 16. In some examples, the layout engine 14 processes the document content 16 on a page-by-page basis. In other examples, the layout engine 14 may process the document content 16 on a multi-page basis. The layout engine 14 may be implemented in any computing environment, including in digital electronic circuitry or in computer hardware, or machine readable instructions (including firmware or software). In some examples, the layout engine 14 is implemented by one or more respective machine readable instructions modules that are executed on a computer. Computer process instructions for implementing the layout engine 14 and the data it generates are stored in one or more computer-readable media.

The document content 16 typically includes a list of document contents (e.g., figures and text), a description of the logical relationships (e.g., linkages) between the document contents, and annotations that assigning respective types to the contents (e.g., title, header, sidebar, pull quote, and advertisement) in accordance with a content description schema that is parsable by the layout engine 14. In an example for a typical magazine, the document content 16 may include a stream of text blocks, a stream of figures, a stream of sidebars, a stream of pull quotes, a stream of advertisements, and a description of the logical relationships between these contents. In some examples, the document content 16 is contained in an XML (Extensible Markup Language) data file that describes the structure of the document contents and identifies the types of the document content elements with attribute tags in accordance with a markup language schema. This representation decouples the content from the presentation and allows schema. This representation decouples the content from the presentation and allows variation in the size, number and relationship among content blocks. Adding or deleting content elements is accomplished easily by addition or deletion of sub-trees in the XML structure, and content modifications simply involve changing the content of an XML leaf-node.
In this example, the XML document content specification identifies a template library (i.e., ABC.xml) and a template collection (i.e., report) within the template library from which the layout engine selects one or more templates that are adapted to accommodate the documents contained in the XML document content specification. The XML document content specification also describes a title, a sidebar that includes a figure (i.e., XYZ.jpg) and a number of text blocks, a header, several paragraphs, a number of list items, and a number of spacers.

FIG. 2 shows an example of a set of 22 of the variable document templates 18, where the s represent template element parameters (e.g., white space dimensions and figure scale factors) that influence the adaptability of the template elements to different document content. The variable document templates 18 are selected from a template library 24 that contains an indexed set of variable document templates. FIG. 3 shows an example of an hierarchical arrangement of templates (T1, ..., TMN) into collections (Collection1, Collection2, ..., CollectionN) of a template library. In some examples, each template library represents a particular type of document (e.g., annual report or newsletter), and each collection represents a section (e.g., appendix or table of contents) of a document of the type represented by the library. In some examples, a
specification of all the templates and collections of a particular template library is stored in one or more XML data files. Different styles can be applied to the same variable document template via the style sheets. A variable document template typically specifies one or more associated style sheets for formatting respective sections of the template. As explained above, the content blocks within the document content typically have attributes that denote their type (e.g., text blocks may be tagged as header, sub-header, attributes that denote their type (e.g., text blocks may be tagged as header, sub-header, list item, paragraph, and caption). The document style sheet 20 associated with the selected variable document template defines the type definitions and the formatting for the style of the content blocks in the document content. For example, the style sheets define a “header” to use Arial bold font with a specified font size and line spacing. Different style sheets can apply different formatting to the same document content. Also, type definitions are scoped within template elements. For example, two different types of sidebars may have different text formatting applied to side-bar text with a sub-header attribute. The style sheet also typically defines overall document characteristics, such as margins, bleeds, page dimensions, and spreads. Different sections of the same document may be formatted with different style sheets.

The document may be any type of electronic document, including an editable markup document (e.g., an Adobe FrameMaker document, a rich text format document, or a Microsoft Word document), a formatted markup document (e.g., a PDF document or an Adobe PostScript document), a rendered image document (e.g., a bit map image document). The document may contain one or more pages of content.

In operation, the layout engine 14 processes the document content to determine an appropriate variable document template 18 to select from the template library 24 for each page of the document. The layout engine 14 also selects one or more style sheets 20 for formatting the document content. Based on the document content, the selected variable document templates 18, and the selected style sheets 20, determines the values of the variables in the selected variable document templates 18 based on an optimization process. The layout engine generates the document based on the variable document templates 18, the determined variable values, and the document content. In this process, the layout determined variable values, and the document content. In this process, the layout
engine 14 populates the variables in the variable document templates 18 with the
determined variable values. At this point, the resulting one or more layout templates 18
can specify the layout of the document. The layout engine 14 generates, the
fully-specified layout of the document 2. The layout engine 14 generates the
fully-specified layout template 18 by rendering the document content 16. In accordance with the resulting,
document 2, each block representing text such as one or more paragraphs or a heading, and the figures portion of the raw input content
is represented by a square labeled 1F-5F, each square representing a figure. The
raw input content also includes semantic information represented by lines extending between text blocks and figures. Semantic information can include which text blocks correspond to headings and which text blocks reference figures. For example, text block 1T makes reference to figure 1F and text block 9T makes reference to figure 5F. Semantic information also identifies which text blocks and figures are associated with a reference. For example, reference 64 corresponds to text block 6T and figure 3F. Note the raw input content does not include which text, figures, and references are to be selected for each page of the overall document, references are to be selected for each page of the overall document, references are to be selected for each page of the overall document.

An allocation of text, figures, and references is determined for each page of the document. Each allocation corresponds to the number of lines of text in the page. Each allocation corresponds to the number of lines of text \( n_t \) and the number of figures \( n_f \) assigned to a respective page of the output document.

Allocation of content to the first \( j \) pages is represented by a random variable \( A \), where \( j \) is a non-negative integer used to identify a page of the document. In the examples illustrated herein, \( j \) equals 0 for the first page of the document. A random variable is a function whose domain is a sample space and whose range is a set of real numbers. For example, referring to FIG. 5, an example of a sample space corresponding to the first page of the document can be the set:

\[
S = \{1T;1F,1T,2T;1F,1T,2T,3T;1F,2F,1T,2T,3T,AT;1F,2F\}
\]

\[
S_0 = \{1T;1F,1T,2T;1F,1T,2T,3T;1F,2F,1T,2T,3T,4F;1F,2F\}
\]
where each element in $S_0$ is a bracket listing text blocks and figures that can be allocated to each page.

The random variable $A_{ij}$ assigns a real, non-negative value to each element in $S_i$, where $S_i$ is a template.

Allocations for pages 2 through $P+1$ are denoted by $A_{ij}$ through $A_{iP}$, respectively, and are similarly defined for a subsequent page. The allocation for the previous page is dependent upon the allocation for the previous page. The allocation for each page is determined through $A_{ij}$.

Below, optimal content allocations $A_0^*$ are determined for each set of pages $[0:i]$. For example, a possible optimal allocation $A_0^*$ for the first page ($P=0$) may be the sample space element $[1T, 2T, 3T, 1F, 2F]$. For the first page, the optimal content allocations $A_0^*$ are determined for each set of pages $[0:1]$.

Once the optimal allocations are determined for the document, an optimal template is determined for each optimal allocation. The optimal template is associated with the original allocation $A_0^*$.

These templates form a sample space of a template library 66 that can accommodate text and figures associated with the optimal allocation $A_0^*$. These templates are stored in the template library 66.

Once an optimal template is determined for each page of the document, an optimal set of template parameters $\Theta$ is associated with the parameters $\Theta_j$ of each subsequent template.

Note that with the exception of the first page, allocations for subsequent pages depend on the allocation for the previous page. For example, consider again the example allocation of text blocks and figures for the first page, $[1T, 2T, 3T, 1F, 2F]$. The allocation for the second page cannot also include text blocks $[1T, 2T, 3T, 1F, 2F]$. The allocation for the second page also include text blocks...
17, 2T, 3T and figures 1F and 2F, because these text blocks and figures have already
been assigned to the first page.

[0033] FIG. 5 shows an example of a method by which the layout engine 14 generates a
document having an optimal number of pages to which document content is allocated. In this process, the layout engine 14 allocates the document content to
is allocated. In this process, the layout engine 14 allocates the document content 16 to
an optimal number of pages that constitute the document 12. In accordance with this
an optimal number of pages that constitute the document 12. In accordance with this
method, the layout engine 14 receives the document content 16 for rendering in
method, the layout engine 14 receives the document content 6 for rendering in
accordance with one or more of the variable document templates 18 (FIG. 5, block 52),
accordance with one or more of the variable document templates 18 (FIG. 5, block 52).
Each variable document template includes one or more respective template elements
Each variable document template includes one or more respective template elements
associated with respective parameters defining relative sizes and positions of the
associated with respective parameters defining relative sizes and positions of the
template elements on a page. The layout engine 14 assigns scores to respective
template elements on a page. The layout engine 14 assigns scores to respective
allocations of the document content to a page (FIG. 5, block 54). Based on the scores,
the layout engine 14 determines an optimal number of pages to layout the document
content (FIG. 5, block 56). The layout engine 14 ascertains from the optimal number
of pages an optimal allocation of the document content to each of the optimal
number of pages, an optimal one of the variable document templates for each of
the optimal number of pages, and optimal values of the parameters of the template
elements of the ascertained optimal variable document template for each of the optimal
number of pages (FIG. 5, block 58). The layout engine 14 generates a document based
on the ascertained optimal allocations of the document content to the optimal number
of pages (FIG. 5, block 60).

[0034] In some examples, for each of the optimal number of pages, the layout
engine 14 determines a respective one of the one or more variable document templates
for rendering the respective optimal allocation of the document content ascertained for
for rendering the respective optimal allocation of the document content ascertained for
the page. As explained above, each of the variable document templates typically
includes values of parameters specifying respective relative positions and sizes for
document content and influencing the adaptability of the template element to different
document contents, where the distribution of parameter values characterizes a degree
development contents, where the distribution of parameter values characterizes a degree
of variability of the associated adaptive layout variable. In these examples, for each of
the optimal number of pages, the layout engine 14 determines optimal values of the
parameters of the respective variable document template respectively determined for
the page template of the respective variable document template respectively determined for
the page.
In some examples, the layout engine 14 is implemented by a probabilistic layout engine that produces variable content documents based on a probabilistic model that assigns a probability score to every possible document, with higher quality documents receiving higher scores. The model parameters typically are given content (e.g., text, figures, and logical relationships between them) by determining given content (e.g., text, figures, and logical relationships between them) by determining a probabilistic inference over a Bayesian network. Examples of the probabilistic layout engine are described in detail below. Additional details regarding the structure and operation of examples of the probabilistic layout engine are described in co-pending International Patent Application No. PCT/US09/61320, filed October 20, 2009, and in co-pending International Patent Application No. PCT/US09/62288, filed October 28, 2009.

The probabilistic layout engine typically includes three components: a parser that parses style sheets, variable document templates, and input document content into internal data structures; an inference engine that computes optimal layouts, given the document content; and a rendering engine that renders the final output document.

In some examples, the parser module includes three separate parsers: a style sheet parser, a content parser, and a variable document template parser. The style sheet parser reads the style sheet 20 for each document content stream 16 and creates a style structure that includes document style and font styles. The content parser reads the content stream and creates respective arrays of structures for figures, text, and sidebars. The text structure array (referred to as the chunk array) includes information about each independent chunk of text that are placed on the page. A single text block in the content stream may be chunked as a whole if it cannot flow across columns or pages (e.g., headings, text within sidebars, etc.). If the text block is allowed to flow (e.g., paragraphs, lists, etc.), it is decomposed into smaller chunks that are rendered atomically. Each structure in the chunk array includes its index in the array, chunk height, whether the column or page breaks are allowed at the chunk, the identifier of the content block to which the chunk belongs, the block type, and an index into the style the content block to which the chunk belongs, the block type, and an index into the style
array to access the style to render the chunk. The height of a chunk is determined by
array to access the style to render the chunk. The height of a chunk is determined by
rendering the text chunk at all possible text widths using the specified style in an offline
rendering process. In one example, the Scribus desktop publishing rendering engine, which allows the probabilistic
layout engine to query the number of rendered lines via an API (Application Programming
Interface) of the Scribus application. The probabilistic layout engine uses the number of
lines and information regarding the font style and line spacing to calculate the rendered
height of a chunk.

Each figure structure in the figure structure array encapsulates the
figure properties of an actual figure in the content stream, such as width, height, source
filename, caption and the text block id of a text block which references the figure. Figure
captions are handled similar to a single text chunk described above allowing various
caption styles, (e.g., full-width captions span text columns, while column width captions span a single text column).

Each content sidebar structure in the sidebar structure array may appear in any sidebar template slot; unless explicitly restricted, so the sidebar array has elements which are themselves arrays with individual elements describing allocations to different possible sidebar styles. Each of these structures has a separate figure array, and chunk array, for figures and text that appear within a particular template sidebar.

Given the document content 16, style sheets 20, and the variable
document templates 18, the inference engine solves for the optimal layout of the given
document template for, the inference engine solves for the optimal layout of the given
content by simultaneously allocating content to a sequence of templates chosen from
content by simultaneously allocating content to a sequence of templates chosen from
the template library, and solving for template parameters that allow maximal page fill
the template library, and solving for template parameters that allow maximal page fill
while incorporating the aesthetic judgements of the graphic designers encoded in the
while incorporating the aesthetic judgments of the graphic designers encoded in the
prior parameter distributions. The inference engine is based on a framework referred to
herein as the Probabilistic Document Model (PDM), which models the generation of
arbitrary "multi-page" documents.

By way of overview, according to the PDM the \( i \) th page of a document
is generated by first choosing a template \( T \) from a library of templates, then setting its
parameter vector \( \Theta \) and then allocating content \( C \) (e.g., images, sidebars, and text blocks) to that page. Each of these tasks is performed by sampling from a conditional
probability distribution. For example, \( P(T_i = t_k) \) is the probability of picking a particular
probability distribution. For example, \( P(T_i = t_k) \) is the probability of picking a particular
template \( t_i \) for the \( i \)th page from a set of \( K \) templates in the template library. If there is no template \( t_i \) for the \( i \)th page, then the template is sampled from a set of \( K \) templates in the template library, with the \( i \)th page being the page that is allocated to the previous template \( t_{i-1} \) with probability \( P(A_i | A_{i-1}, \Theta) \). Other values can be set for the template probabilities so that certain templates can be more or less likely to be used in creating a document. Thus:

\[
P(T_i) = p_k \quad \text{where} \quad \sum_k p_k = 1
\]

[0043] Once a template is selected, the inference engine samples its parameter vector \( \Theta \) from the conditional distribution \( P(\Theta | T_i) \). The inference engine models this distribution to be a multivariate normal distribution that describes the prior probabilities of various parameter settings. This distribution can be directly derived from the parameter distributions in the variable document templates that encode the template designer's aesthetic choices about desired parameter values. Sampling from this distribution makes these aesthetic parameter settings more likely. Thus:

\[
P(\Theta | T_i) = N(\bar{\Theta}_i | \Lambda^{-1})
\]

Where \( \bar{\Theta}_i \) and \( \Lambda \) represent the means and precisions of the parameters entered by a graphic designer. In fact, any desired linear relation (proportions, ratios, etc.) between parameters can be captured in the above prior probability distribution.

[0044] Finally, the inference engine samples the allocation for the current page \( C_i \) from a probability distribution \( P(C_i | T_i, \Theta) \). In this example, the model assumes that the current allocation is independent of the template and parameter settings for all previous pages, given the allocation to the previous page and the page settings for the current page. The allocation to the previous page affects the probability of an allocation for the current page via the logical relationship content in the probability of an allocation for the current page via the logical relationship content in the previous page. The algorithm was used to infer the probability of content on the current page being allocated to the previous page. First, information on content on the previous page has been sufficient to conclude that all prior content has already been allocated. Thus, the previous page allocation allows the inference engine to determine if a figure or sidebar appearing on the current page is referenced in a prior page. This would reduce the probability of the current allocation since the figure/sidebar does not appear on the same page as its reference. Furthermore, \( P(A_i | A_{i-1}, \Theta) \) depends on the parameter settings since certain parameter settings fill the page with different contents.
the existing content better than others and are hence assigned higher probability. This conditional distribution is given by:

\[
P(C_i | C_{i-1}, \Theta_i) \propto \exp(-y | R(C_i, C_{i-1})) \prod_{m=1}^{M} N(H_m(C_i) | \Theta_m^y, \alpha^{-1}) \prod_{n=1}^{N} N(W_n(C_i) | \Theta_n^x, \beta^{-1})
\]

[0045] In the above equation, \(| R(A_i, A_{i-1}) |\) represents the number of dangling references due to the allocation \(A_i\) to the \(i^{th}\) page and \(A_{i-1}\) to the \((i-1)^{th}\) page. The constant \(y\) represents an exponential weighting factor that represents how much to penalize mismatched references in the probability distribution. \(H_m(A_i)\) and \(W_n(A_i)\) represent the height and width of a \(Y\) and \(X\) path group respectively after subtracting fixed spaces and reserved regions in the \(Y\) and \(X\) directions. The variables \(y_m\) and \(x_n\) represent content-dependent vectors that represent actual figure heights, widths and whitespace in each direction, for each path group. For good page fill in the \(Y\) and \(X\) directions \(H_m(A_i) - \Theta_T^y y_m \approx 0\) and \(W_n(A_i) - \Theta_T^x x_n \approx 0\). The normal distributions above simply assign high probability to these events and lower probability for deviations from ideal. The constants \(a_i\) and \(\beta_i\) are precision variables of the normal distribution that control the degree to which production of full pages is favored.

[0046] The following is a detailed description of examples of processes for generating a document from one or more variable document templates in accordance with the Probabilistic Document Model described above.

[0047] In these examples, the probabilistic layout engine generates documents by repeating the sampling process, page by page. Thus, a document is a sequence of templates, template parameters, and content allocations that is sampled from an overall probability distribution over \(P\) pages, given by:

\[
P\left(\{T_j\}, \{\Theta_j\}, \{C_j\}\right) = P(A_0 | \Theta_0) P(\Theta_0 | T_0) P(T_0) \prod_{i=1}^{P} P(A_i | A_{i-1}, \Theta_i) P(\Theta_i | T_i) P(T_i)
\]

[0048] This distribution may be represented by directed graph shown in FIG. 6. The distribution is generated from the graph by multiplying the conditional probability distributions of each node conditioned on only its parents. Such a model is called a Bayesian network \(\mathcal{N}\) for the underlying probability distribution characterizing conditional independencies of the random variables associated with allocations \(A_j\), templates \(T_j\) and content allocations. Such a model is called a Bayesian network \(\mathcal{N}\) for the underlying probability distribution characterizing conditional independencies of the random variables associated with allocations \(A_j\), templates \(T_j\), and content allocations. Such a model is called a Bayesian network \(\mathcal{N}\) for the underlying probability distribution characterizing conditional independencies of the random variables associated with allocations \(A_j\), templates \(T_j\), and content allocations.
and parameters $\Theta_j$. Each node of the Bayesian network 70 represents a random variable corresponding to events in a sample space. For example, node 72 represents a random variable $A$ associated with a sample space of allocations for the first page. Node 74 represents a random variable $T$, associated with a sample space of templates for the allocation selected for the first page, and node 76 represents the random variable $\Theta_0$ associated with a sample space of parameters for the template selected for the first page. Directional arrows in the Bayesian network independently identify conditional probabilities between nodes. For example, directional arrow $78$ represents the conditional probability $P(\Theta_0|\Theta_0)$ for a set of parameters $\Theta_0$ given a template $T_0$, but the allocations $A_1$, $A_p$ have more than one parent node. Thus, the conditional probability for the node 80 representing the allocation $A_1$ is $P(A_1|A_0, \Theta_0)$.

Note that the Bayesian network defines an conditional independence structure. In other words, any node is conditionally independent of its non-descendent given its parent.

For nodes like $T_0$, $T_p$, the probabilities associated with these modes $P(T_0)$, $P(T_p)$ are not conditioned on any other nodes.

The Probabilistic Document Model is generative, in the sense that a sequential process (as described above) can be used to generate pages and thus documents from the model. The documents generated by this process are samples drawn by the inference engine from the probability distribution described above. Although the sampling procedure generates documents with various probabilities (recall that higher probabilities translate to higher quality), the inference engine is configured to find the document that has the highest probability. In this process, the inference engine finds the document that has the highest probability. In this process, the inference engine computes the optimizing sequence of templates $\{A_i\}$, template parameters $\{\Theta_i\}$, and computes the optimizing sequence of templates $\{A'_i\}$, template parameters $\{\Theta'_i\}$, and

A joint probability distribution that characterizes the conditional probabilities of a Bayesian network is a product of the probabilities of the parent nodes and the conditional probabilities. Thus, the joint probability distribution associated with the Bayesian network 70 is given by:

$$P(\Theta_0, T_0, A_0, A_1, \ldots, A_p) = P(\Theta_0|\Theta_0)P(T_0|\Theta_0)P(A_0|A_0, \Theta_0)P(A_1|A_0, \Theta_0)\cdots P(A_p|A_{p-1}, \Theta_0)$$
As shown in FIG. 6, allocation $A_0$ for the first page "0" is independent, but allocations for each of the subsequent pages depend on the allocation for the previous page. Thus, $P\left(\{T_j\}, \{\Theta_j\}, \{A_j\}\right)$ includes the terms $P\left(A_j | A_{j-1}, \Theta_j\right)$. 

In the examples described herein, the optimization process performed by the inference engine is predicated on maximizing $P\left(\{T_j\}, \{\Theta_j\}, \{A_j\}\right)$ with the assumption that the larger the probability $P\left(\{T_j\}, \{\Theta_j\}, \{A_j\}\right)$, the closer the document layout is to having the following desired document properties:

(i) each page of the document should look as good as possible to achieve overall optimal layout quality;

(ii) text blocks that reference figures and the corresponding figures should appear on the same page; and

(iii) the total number of pages is minimized.

In order to determine the sets $\{T_j\}, \{\Theta_j\},$ and $\{A_j\}$ for a document that give the maximum probability, a maximum joint probability distribution is defined as follows:

$$P^*\left(\{T_j\}, \{\Theta_j\}, \{A_j\}\right) = \max_{\{A_j\}} \phi \left(A_0, \prod_{j=1}^P \phi \left(A_j, A_{j-1}\right) \right) \tag{7}$$

$$\phi \left(A_j, A_{j-1}\right) = \max_{T_j} \psi \left(A_j, A_{j-1}, T_j\right) P\left(T_j\right) \tag{8}$$

$$\psi \left(A_j, A_{j-1}, T_j\right) = \max_{\Theta_j} P\left(A_j | A_{j-1}, \Theta_j\right) P\left(\Theta_j | T_j\right) \tag{9}$$

Equations (7), (8), and (9) are used to determine optimal allocations, templates, and template parameters. As explained in detail below, the set of allocations $\{A_j\}$ that maximize equation (7) can be obtained by first determining the $\phi$s. Each $\phi$ is a function of random variables, and is the maximum of a sequence of real numbers, one for each template $T_j$, as described in equation (8). Hence for each $A_j$ and $A_{j-1}$ we have a...
maximizing template $t^*_j$. Thus, the $\phi$'s assign a respective maximal score for each
maximizing template $t^*_j$. Thus, the $\phi$'s assign a respective maximal score for each
layout of the respective allocation on the page in accordance with a respective one of
the respective allocation on the page in accordance with a respective one of
layout, of the respective allocation on the page in accordance with a respective one of
variable document templates. For the first page, $\phi(A_0)$ is the maximum of the range
of real values associated with the allocation $A_0$. For subsequent pages, $\phi(A_j, A_{j-1})$ is the
real values associated with the allocation $A_j$. For subsequent pages, $\phi(A_j, A_{j-1})$ is the
maximum of the range of real values associated with the allocations $A_j$ and $A_{j-1}$.

[0054] As explained above, although all the variables influence each other in
general, a variable is directly influenced by a few neighbors at most. This fact allows
the inference engine to use the structure of the graph in performing the desired
maximization. The variable $\Theta_0$ appears in only two terms with two other variables, $A_0$
maximization. The variable $\Theta_0$ appears in only two terms with two other variables, $A_0$
and $T_0$. This allows the inference engine to maximize over $\Theta_0$ without looping over
every possible value of every node (variable) in the network. Thus we have:

$$\Psi(A_0, T_0) = \max_{\Theta_0} P(A_0 | \Theta_0) P(\Theta_0 | T_0) = \max_{\Theta_0} P(\Theta_0 | A_0, T_0)$$

(10)

[0055] This maximization is performed for all possible values of $\Theta_0$ and $T_0$, but
the inference engine does not need to consider variations in other variables in the
network. Even so, this is still a challenging task in general. The particular
distributions that are used for $P(\Theta_0 | T_0)$ and $P(A_j | A_{j-1}, \Theta_0)$ make this task particularly
efficient for the inference engine since these are conjugate normal distributions and
their product also is normal. The maximizing value is the mean of the distribution, which
can be computed in closed form using the equation
can be computed in closed form using the equation

$$\Theta_0^* = \left( A + \sum_m \alpha x_m x^*_m + \sum_n \beta y_n y^*_n \right)^{-1} \left( \Lambda \Theta + \sum_m \alpha W_m (A_j) x_m + \sum_n \beta H_n (A_j) y_n \right)$$

(11)

[0056] This efficient to compute, and the effect of incremental allocations is
even more efficient to compute. Allocating an additional text chunk would affect
the right hand side of the above equation and a few multiply accumulates are used to
update from $\Psi(A_0, T_0)$ to $\Psi(A_0, T_1)$. In some cases additional speedup in
computing $\Psi(A_0, T_1)$ is obtained by configuring the inference engine to:
screen allocation of $A_0$ and avoiding computation altogether for invalid allocations (e.g., figure or
sidebar occurring before its reference, widowed orphaned text chunks, etc.); screen
templates for compatibility (e.g., no content with two figures cannot be allocated to a
template with only one figure); and screen for too much and too little content that make


it impossible to fill the page even when template parameters are maximum or minimum.
These screening approaches significantly reduce the number of cases for which
it impossible to fill the page even when template parameters are maximum or minimum.

Once \((A_0, T_0)\) is computed, the inference engine maximizes over

\[ \phi(A_0) = \max_{T_0} \Psi(A_0, T_0) P(T_0) \]  

In general, eliminating (via maximization) variables \( \Theta \) and \( \Upsilon \) yields the functions \( \phi(A_i, A_{i-1}) \). The functions \( \phi(A_i, A_{i-1}) \) assign scores to respective allocations \( A_i \) of the
document content to a page, where \( A_i \) represents the difference between the
document content allocated to page \( i \) and the document content
respectively allocated to page \( i-1 \). The respective portions of the document content
represented by \( A_i \) are used to determine the best scoring allocation of content
\( A_{i-1} \) to page \( i \). Note that if \( i = 0 \), \( \phi(A_0) \) is simply \( \phi(A_0, [0]) \).

The inference engine proceeds by determining the optimal allocation of
the total content over an optimal number of pages (i.e., the number of pages, that
maximizes the overall score for the document).

FIG. 7 shows an example of a method by which the inference engine
determines the optimal number of pages and generates a data structure (e.g., a table)
that stores a sequence of optimal allocations of the document content to respective sets
of the pages.

In accordance with the method of FIG. 7, the inference engine selects
the next cumulative subset of document content \( A \) (FIG. 7, block 90). In some
elements, the next cumulative subset of document content is represented by \(|A|\), which
ranges from \(|A| = 1\) to \(|A| = |C|\), where \( C \) represents the total document content to be
allocated.

The inference engine determines a highest scoring joint combination of
a current allocation of content within the current cumulative subset to a current page
and previous allocation of content within the next cumulative subset to each page
preceding the current page (FIG. 7, block 92). In some elements, the score of the joint
combination is given by

\[ P^* (E \phi(A_i, B) P^*(B)) \]  

where \( E \phi(A_i, B) \) and \( P^*(B) \) is the maximum score associated with the prior allocation \( B \), the highest scoring joint

maximum score associated with the prior allocation \( B \), the highest scoring joint
combination corresponds to the allocation of A-B to the current page and the allocation combination corresponds to the allocation of A-B to the current page and the allocation of the remaining content B to each of the pages preceding the current page that maximize the joint combination score. In some examples, in the process of determining the optimal number of pages, the inference engine performs the following operations for each of multiple cumulative subsets of the document content: the layout engine determines a highest scoring combination of a current allocation of content within the cumulative subset to a current page and a prior allocation of content within the cumulative subset to each page preceding the current page; and the inference engine sets a current page count equal to a page count associated with the current page. In these examples, the optimal number of pages corresponds to the current page count set for the cumulative subset equal to all the document content.

[0062] The inference engine stores the previous allocation in a data structure in association with the page preceding the current page (FIG. 7, block 94). In some examples, the inference engine stores the prior allocation B that maximizes the joint combination score \( P^{nk} \in \phi(A,B) \). In some examples, for each of multiple cumulative subsets of the document content, the inference engine stores the prior allocation of the document content in a data structure in association with an index value corresponding to the page preceding the current page.

[0063] The inference engine sets a current page count to a single page count higher than a page count associated with the page preceding the current page (FIG. 7, block 98). That is, the count of the pages preceding the current page is incremented by one to produce the current page count.

[0064] The process is repeated for each cumulative subset of document content until current cumulative subset is equal to the total document content to be allocated (FIG. 7, block 98). The process is repeated for each cumulative subset of document content.

[0065] Once the optimal number of pages has been determined, the inference engine ascertains the optimal allocation of the document content to each of the optimal number of pages. In some examples, this process involves allocating all the document content to at least one of the pages highest in page number and, for each pair of consecutive pages from at least one of the pages to a lower one of the pages, assigning to the page of the pair higher in page number a respective portion of the document content corresponding to a difference between the document content respectively allocated to it and the page of the pair higher in page number and the document content respectively to the page of the pair higher in page number and the document content respectively...
allocated to the page of the pair consecutively lower in page number. In examples in which the prior allocation of the document content are stored in a data structure in association with an index value corresponding to the page preceding the current page, the process of assigning the respective portion of the document content to each page may involve sequentially retrieving the prior allocations stored in the data structure in order from the highest index value to the lowest index value, and determining respective differences between consecutive ones of the retrieved prior allocations.

Once the allocations for each page are determined, the inference engine looks up the optimal template for each page by finding the template T* that maximizes $\Psi(C^*, T)$. Once the template and the allocation are known, the inference engine determines the template parameters using the closed form solution for $\Theta^*$ given above. Once the page count, allocations to pages, template selection, and parameter estimation are done, the inference task is complete. The solution found by this approach is globally optimal, attaining a significant computational advantage over exhaustive enumeration of all possibilities.

After the set of optimal allocations have been determined for each optimal allocation, equations (8) and (9) can be used to determine the optimal $T_j$ and $\Theta_j$.

For each $A_j$, there is a set of $T_j$'s. Once $\Phi(A_j | A_{j-1})$ is determined, the corresponding $T_j$ maximizes equation (8), and the corresponding template parameters $\Theta_j$ maximize equation (3). In equation (9), $P(A_j | A_{j-1}, \Theta_j)$ is the product of layout quality, reference quality, and page qualities probabilities given by:

$$P(A_j | A_{j-1}, \Theta_j) = P_L(A_j | A_{j-1}, \Theta_j) P_R(A_j | A_{j-1}) P_P(A_j | A_{j-1})$$

(13)

The conditional probability $P_L(A_j | A_{j-1}, \Theta_j)$ associated with layout quality is determined by a document designer. The reference quality probability can be defined as follows:

$$P_R(A_j | A_{j-1}, \Theta_j) = \exp \left(-\gamma |R_{j-1,j} - 1| \right)$$

(14)

where $\gamma$ is a reference constant assigned by the document designer, and $|R_{j-1,j}$ corresponds to the number of misplaced references due to the previous allocation $A_{j-1}$.

The page quantity probability can be defined as follows:

$$P_P(A_j | A_{j-1}) = \exp \left(-\delta \right)$$

(15)
δ is a page constant assigned by the document designer and corresponds to a page number penalty that is used to control the overall number of pages in the final document.

Next, a closed form equation for determining the parameter vector $\Theta_j$ for each template is now described. This closed form description can be obtained by considering the relationship between dimensions of elements of a template with $m$ rows of image fields and $n$ columns of image fields and the corresponding parameter vector $\Theta$ in terms of Bayes' Theorem from probability theory as follows:

$$P(\Theta | \tilde{W}, \tilde{H}, \tilde{x}, \tilde{y}) \propto P(\tilde{W}, \tilde{H}, \tilde{x}, \tilde{y} | \Theta) P(\Theta)$$

Equation (16) is in the form of Bayes' Theorem but with the normalizing probability $P(\tilde{W}, \tilde{H}, \tilde{x}, \tilde{y})$ excluded from the denominator of the right-hand side of equation (1). For certain examples, the prior probability can be characterized by a normal, or Gaussian, probability distribution given by:

$$P(\Theta) \approx N(\Theta; \Lambda^{-1}) \propto \exp\left(\frac{1}{2} (\Theta - \tilde{\Theta})^T \Lambda^{-1} (\Theta - \tilde{\Theta})\right)$$
where $\Theta_1$ is a vector composed of independent mean values for the parameters set by a document designer; $\Lambda_1$ is a diagonal matrix of variances for the independent parameters set by the document designer; $\Lambda_2 = C^T \Delta C$ is a non-diagonal covariance matrix for dependent parameters; and $\Theta_2 = A^T C \Delta^T A d$ is a vector composed of dependent mean values for the parameters. The matrix $C$ and the vector $d$ characterize the linear relationships between the parameters of the parameter vector $\Theta$ and $\Delta$ is a covariance precision matrix.

Returning to equation (16), the term $P(\hat{\Theta}, \hat{x}, \hat{y}|\hat{\Theta})$ is the conditional probability of an event composed of $\hat{W}$, $\hat{H}$, $\hat{x}$, and $\hat{y}$, given the occurrence of the parameters of the parameter vector $\Theta$. In certain examples, the term $P(\hat{W}, \hat{H}, \hat{x}, \hat{y}|\hat{\Theta})$ can be characterized as follows:

$$P(\hat{W}, \hat{H}, \hat{x}, \hat{y}|\hat{\Theta}) \propto \prod_i \prod_j N(W_i | \Theta^T \hat{x}_i, \alpha_i^{-1}) N(H_j | \Theta^T \hat{y}_j, \beta_j^{-1})$$

(18)

where

$$N(W_i | \Theta^T \hat{x}_i, \alpha_i^{-1}) \propto \exp\left(-\frac{\alpha_i}{2} (\Theta^T \hat{x}_i - W_i)^2\right)$$

and

$$N(H_j | \Theta^T \hat{y}_j, \beta_j^{-1}) \propto \exp\left(-\frac{\beta_j}{2} (\Theta^T \hat{y}_j - H_j)^2\right)$$

are normal probability distributions. The variables $\alpha_i^{-1}$ and $\beta_j^{-1}$ are variances and $\alpha_i^{-1}$ and $\beta_j^{-1}$ are variances and $W_i$ and $H_j$ represent mean values for the distributions $N(W_i | \Theta^T \hat{x}_i, \alpha_i^{-1})$ and $N(H_j | \Theta^T \hat{y}_j, \beta_j^{-1})$, respectively. Normal distributions can be used to characterize, at least approximately, the probability distribution of a variable that tends to cluster around the mean. In other words, variables close to the mean are more likely to occur than those farther from the mean. The normal distributions $N(W_i | \Theta^T \hat{x}_i, \alpha_i^{-1})$ and $N(H_j | \Theta^T \hat{y}_j, \beta_j^{-1})$, respectively, characterize the probability distributions of the variables $W_i$ and $H_j$ about the mean values $\Theta^T \hat{x}_i$ and $\Theta^T \hat{y}_j$, respectively.
The posterior probability $P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y})$ can be maximized when the exponents of the normal distributions of equation (8) satisfy the following conditions:

$$\mathbf{x}_i - \mathbf{W}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{H}_i)$$

for all $i$ and $j$. As described above, for a template, $\mathbf{W}_i$ and $\mathbf{H}_i$ are constants and the elements of $\mathbf{x}_i$ and $\mathbf{y}_j$ are constants. These conditions are satisfied by determining a parameter vector $\mathbf{\theta}^{MAP}$ that maximizes the posterior probability $P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y})$. The parameter vector that maximizes the posterior probability $P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y})$ as a multivariate normal distribution with a well-characterized mean and variance as follows:

$$P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y}) = N\left(\mathbf{\theta}^{MAP}, \mathbf{A}^{-1}\right)$$

The parameter vector $\mathbf{\theta}^{MAP}$ is the mean of the normal distribution characterization of the posterior probability $P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y})$, and $\mathbf{\theta}$ maximizes $P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y})$ when $\mathbf{\theta}$ equals $\mathbf{\theta}^{MAP}$. Solving $P(\mathbf{\theta}|\mathbf{W}, \mathbf{H}, \mathbf{x}, \mathbf{y})$ for $\mathbf{\theta}^{MAP}$ gives the following closed form expression:

$$\mathbf{\theta}^{MAP} = \left(A + \sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i^T + \sum_j \beta_j \mathbf{y}_j \mathbf{y}_j^T\right)^{-1}\left(A\mathbf{\theta} + \sum_i \alpha_i \mathbf{W}_i \mathbf{x}_i + \sum_j \beta_j \mathbf{H}_j \mathbf{y}_j\right)$$

The parameter vector $\mathbf{\theta}^{MAP}$ can also be rewritten in matrix from as follows:

$$\mathbf{\theta}^{MAP} = A^{-1}\mathbf{b}$$

where $A = \mathbf{A} + \sum_i \alpha_i \mathbf{x}_i \mathbf{x}_i^T + \sum_j \beta_j \mathbf{y}_j \mathbf{y}_j^T$ is a matrix and $A^{-1}$ is the inverse of $A$, and

$$\mathbf{b} = A\mathbf{\theta} + \sum_i \alpha_i \mathbf{W}_i \mathbf{x}_i + \sum_j \beta_j \mathbf{H}_j \mathbf{y}_j$$

is a vector.

In summary, given a single page template and figures to be placed in the image fields of the template, the parameters used to scale the images and white spaces of the template can be determined from the closed form equation for $\mathbf{\theta}^{MAP}$.

Once the parameters of the parameter vector $\mathbf{\theta}^{MAP}$ are determined using the closed form equation for $\mathbf{\theta}^{MAP}$, the template can be rendered by multiplying using the closed form equation for $\mathbf{\theta}^{MAP}$, the template can be rendered by multiplying
un-scaled dimensions of the images and widths of the white spaces by corresponding
parameters of the parameter vector
\theta_{MAP}^{(0)} \] The elements of the parameter vector \( \theta_{MAP}^{(0)} \) may also be subject to
boundary conditions on the image fields and white space dimensions arising from the
minimum width constraints for the margins. In other examples, in order to determine
\( \theta_{MAP}^{(0)} \) subject to boundary conditions, the vectors \( \bar{x}_1, \bar{x}_2, \bar{y}_1, \) and \( \bar{y}_2 \), the variances \( \alpha_1 \), \( \alpha_2 \), \( \beta_1 \), and \( \beta_2 \), and the constants \( W_1, W_2, H_1, \) and \( H_2 \) are inserted into the linear
\[ \alpha_1, \beta_1, \alpha_2, \beta_2, \] and the constants \( I, W, H \) are inserted into the linear
equation \( \theta_{MAP}^{(0)} = B \), and the matrix equation solved numerically for the parameter
vector \( \theta_{MAP}^{(0)} \) subject to the boundary conditions on the parameters of \( \theta_{MAP}^{(0)} \). The matrix
equation \( \alpha_{MAP}^{(0)} = B \) can be solved using any well-known numerical method for solving
matrix equations subject to boundary conditions on the vector \( \theta_{MAP}^{(0)} \), such as the
conjugate gradient method.

[0076] As explained above, in some examples, the probabilistic layout engine
uses an API to a graphics design or rendering tool (e.g., the Scribus API) to create the
final document. For each page, the rendering engine traverses the Y and X paths of
the chosen page template and renders content blocks as they are encountered.

Optimal template parameters determined via inference are used directly to determine
the positions, widths, and heights of all blocks. The rendering engine produces a renderable
structure that keeps track of items that have already been rendered, to avoid
duplicate rendering. Text blocks are rendered in the Y-path and image blocks are
copied during rendering. Text blocks are rendered in the X-path and image blocks are
rendered after a complete X and Y pass. Separate layers are created to render
background and foreground items.

[0077] Examples of the document processing system 10 may be implemented
by one or more discrete modules (or data processing components) that are not limited
to any particular hardware, or machine readable instructions (e.g., firmware or software)
computer hardware, device driver, or machine readable instructions (including firmware
or software), and in some examples, the functionalities of the modules are combined into a
single data processing component. In some examples, the respective functionalities of
data processing component.
each of one or more of the modules are performed by a respective set of multiple data processing components. The modules of the document processing system 10 may be co-located. In a single apparatus or they may be distributed across multiple apparatus, if located on a single apparatus or they may be distributed across multiple apparatus; if distributed across multiple apparatus, these modules may communicate with each other over local wired or wireless connections, or they may communicate over global network connections (e.g., communications over the Internet).

In some implementations, process instructions (e.g., machine-readable code, such as computer software) for implementing the methods that are executed by the examples of the document processing system 10 are stored in one or more machine-readable media. Storage devices suitable for storing instructions and data include all forms of non-volatile computer-readable memory, including, for example, semiconductor memory devices, such as EPROM, EEPROM, and flash memory devices, magnetic disks such as internal hard disks and removable hard disks, magneto-optical disks, DVD-ROM/RAM, and CD-ROM/RAM.

In general, examples of the variable document template generation system 0 may be implemented in any one of a wide variety of electronic devices, including desktop computers, workstation computers, and server computers.

FIG. 8 shows an example of a computer system 140 that can implement any of the examples of the document processing system 10 that are described herein. The computer system 140 includes a processing unit 142 (CPU), a system memory 144, and a system bus 146 that couples processing unit 142 to system memory 144, and a system bus 146 that couples processing unit 142 to the various components of the computer system 140. The processing unit 142 typically includes one or more processors, each of which may be in the form of any one of various commercially available processors. The system memory 144 typically includes a read only memory (ROM) that stores a basic input/output system (BIOS) that contains start-up routines for the computer system 140 and a random access memory (RAM). The system bus 146 may be a memory bus, a peripheral bus or a local bus, and may be compatible with any of a variety of bus protocols, including PCI, VESA, MicroChannel, ISA, and EISA. The computer system 140 also includes a persistent storage memory 148 (e.g., a hard drive, a floppy drive, a CD-ROM drive, magnetic tape drives, flash memory devices, and digital video disks) that is connected to the system bus (146 and memory devices, and digital video disks) that is connected to the system bus (146 and
contains one or more computer-readable media disks that provide non-volatile or persistent storage for data, data structures, and computer-executable instructions. A user may interact (e.g., enter commands or data) with the computer using one or more input devices (e.g., a keyboard, a computer mouse, a microphone, joystick, and touchpad). Information may be presented through a user interface that is displayed to a user on the display 151 (implemented by, e.g., a display monitor), which is controlled by a display controller 154 (implemented by, e.g., a video graphics card). The computer system 140 also typically includes peripheral output devices, such as speakers and a printer. One or more remote computers may be connected to the computer system 140 through a network interface card (NIC) 156.

As shown in FIG. 8, the system memory 144 also stores the document processing system 10, a graphics driver 158, and processing information 160 that includes input data, processing data, and output data. In some examples, the document processing system 10 interfaces with the graphics driver 158 to present a user interface on the display 151 for managing and controlling the operation of the document processing system 10.

Other embodiments are within the scope of the claims.
CLAIMS

1. A method, comprising:
receiving document content (16) for rendering in accordance with one or more variable document templates (18), wherein each variable document template comprises one or more respective template elements associated with respective parameters defining relative sizes and positions of the template elements on a page;
assigning scores to respective allocations of the document content (16) to a page;
based on the scores, determining an optimal number of pages to layout the document content (16); ascertaining from the optimal number of pages a respective optimal allocation of the document content (16) to each of the optimal number of pages, a respective optimal one of the variable document templates for each of the optimal number of pages, and optimal values of the parameters of the template elements of the ascertained optimal variable document template for each of the optimal number of pages; and generating a document (12) based on the optimal allocations, the optimal variable document templates, and the optimal values of the parameters that respectively are ascertained for the optimal number of pages;
wherein the receiving, the assigning, and the determining are performed by a computer system (140).

2. The method of claim 1, wherein the assigning comprises, for each of the respective allocations, assigning a respective maximal score for a layout of the respective allocations, assigning a respective maximal score for a layout of the respective allocation on a page in accordance with each of the one or more variable document templates (18).

3. The method of claim 1, wherein the determining comprises for each of multiple cumulative subsets of the document content (6):
determining a highest scoring combination of a current allocation of document content within the cumulative subset to a current page and a prior allocation of document content within the cumulative subset to each page preceding the current page; and
setting a current page count equal to a page count associated with the current page; and
wherein the optimal number of pages corresponds to the current page count set for the cumulative subset equal to all the document content (16).

4. The method of claim 3, wherein the setting comprises increment the current page count equal to one more than a count of the pages preceding the current page.

5. The method of claim 3, wherein the ascertainment comprises:

allocating all the document content (16) to a last one of the pages highest in page number; and

for each pair of consecutive pages from a last one of the pages to a first one of the pages, assigning to the page of the pair higher in page number a respective portion of the document content (16) corresponding to a difference between the document content respectively allocated to the page of the pair consecutively lower in page number.

6. The method of claim 5, wherein the determining comprises, for each of multiple cumulative subsets of the document content (16), storing the prior allocation of the document content in a data structure in association with an index value corresponding to the page preceding the current page; and the assigning comprises sequentially retrieving the prior allocations stored in the data structure in order from the highest index value to the lowest index value, and determining respective differences between consecutive ones of the retrieved prior allocations.

7. The method of claim 5, wherein the ascertainment comprises:

for each of the optimal number of pages, determining a respective one of the one or more variable document templates (18) for rendering the respective optimal allocation of the document content (16) ascertained for the page in accordance with a probabilistic optimization process.

The method of claim 6, wherein the ascertainment comprises:

determining respective ones of the variable document templates (18) for the page in accordance with a probabilistic optimization process.
8. The method of claim 7, wherein the ascertaining comprises:
   for each of the optimal number of pages, determining optimal values of the parameters of the respective variable document template respectively determined for the page.

9. The method of claim 1, wherein each variable document template comprises at least one template element that is associated with a respective parameter characterized by one or more values that define a respective degree of variability or the characterized by one or more values that define a respective degree of variability of the associated parameter.

10. Apparatus, comprising:
    a computer-readable memory (144, 148) storing computer-readable instructions; and
    a data processor (142) coupled to the memory, operable to execute the instructions, and based at least in part on the execution of the instructions operable to perform operations comprising:
        receiving document content (16) for rendering in accordance with one or more variable document templates (18), wherein each variable document template comprises one or more respective template elements associated with respective parameters defining relative sizes and positions of the template elements on a page;
        assigning scores to respective allocations of the document content (16) to a page, a respective optimal one of the variable document templates for each of the optimal number of pages, and optimal values of the parameters of the template elements of the ascertained optimal variable document template for each of the optimal number of pages, optimal number of pages;
        based on the scores, determining an optimal number of pages to layout the document content (16), and
        ascertaining from the optimal number of pages a respective optimal allocation of the document content (16) to each of the optimal number of pages; and

generating a document (12) based on the optimal allocations, the optimal variable document templates, and the optimal values of the parameters that respectively are ascertained for the optimal number of pages.

11. The apparatus of claim 10, wherein in the assigning comprises, for each of the respective allocations, assigning a respective maximal score for a layout of the respective allocations, assigning a respective maximal score for a layout of the respective allocation on a page in accordance with each of the one or more variable document templates (18).

12. The apparatus of claim 10, wherein in the determining the data processor is operable to perform operations comprising

for each of multiple cumulative subsets of the document content (16):

determining a highest scoring combination of a current allocation of document content within the cumulative subset to a current page and a prior allocation of document content within the cumulative subset to each page preceding the current page; and

setting a current page count equal to a page count associated with the current page; and

wherein the optimal number of pages corresponds to the current page count set for the cumulative subset equal to all the document content (16).

13. The apparatus of claim 12, wherein in the ascertaining the data processor is operable to perform operations comprising:

allocating all the document content (16) to a last one of the pages highest in page number; and

for each pair of consecutive pages from a last one of the pages to a first one of the pages, assigning to the page of the pair higher in page number a respective portion of the document content (16) corresponding to a difference between the document content respectively allocated to the page of the pair higher in page number and the document content respectively allocated to the page of the pair consecutively lower in page number.
14. The apparatus of claim 3, wherein in the determining the data processor is operable to perform operations comprising, for each of multiple cumulative subsets of the document content stored in a data structure in association with an index value corresponding to the page preceding the current page, and the assigning comprises sequentially retrieving the prior allocations stored in the data structure in order from the highest index value to the lowest index value, and determining respective differences between consecutive ones of the retrieved prior allocations.

15. At least one non-transitory computer-readable medium (144, 148) having computer-readable program code embodied therein, the computer-readable program code adapted to be executed by a computer (140) to implement a method comprising:

receiving document content (16) for rendering in accordance with one or more variable document templates (18), wherein each variable document template comprises one or more respective template elements associated with respective parameters defining relative sizes and positions of the template elements on a page;

assigning scores to respective allocations of the document content (16) to a page;

based on the scores, determining an optimal number of pages to layout the document content (16);

ascertaining from the optimal number of pages a respective optimal allocation of the document content (16) to each of the optimal number of pages, a respective optimal one of the variable document templates for each of the optimal number of pages, and optimal values of the parameters of the template elements of the ascertained optimal variable document template for each of the optimal number of pages; and

generating a document (12) based on the optimal allocations, the optimal variable document templates, and the optimal values of the parameters that respectively are ascertained for the optimal number of pages.
FIG. 3
Receive Document Content For Rendering In Accordance With One Or More Variable Document Templates

Assign Scores To Respective Allocations Of The Document Content To A Page

Based On The Scores, Determine An Optimal Number Of Pages To Layout The Document Content

Ascertain From The Optimal Number Of Pages An Optimal Allocation Of The Document Content To Each Of The Optimal Number Of Pages, A Respective Optimal One Of The Variable Document Templates For Each Of The Optimal Number Of Pages, And Optimal Values Of The Parameters Of The Template Elements Of The Ascertained Optimal Variable Document Template For Each Of The Optimal Number Of Pages

Generate A Document Based On The Ascertained Optimal Allocations Of The Document Content To The Optimal Number Of Pages

FIG. 5

FIG. 6
INTERNATIONAL SEARCH REPORT

International application No.
PCT/US20 10/053890

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/21(2006.01)i, G06F 3/14(2006.01)i, G06F 19/00(2011.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 17/21; G06F 15/00; G06F 17/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords :document,layout,template

C. DOCUMENTS CONSIDERED TO BE RELEVANT

<table>
<thead>
<tr>
<th>Category</th>
<th>Citation of document, with indication, where appropriate, of the relevant passages</th>
<th>Relevant to claim No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>US 2009-0021297 A1 (BARGHEER, DAVID et al.) 24 January 2008</td>
<td>1,9-10,15</td>
</tr>
<tr>
<td></td>
<td>See abstract, descriptions [0024]-[0032] and [0076]. Figures 9A-9C, and claims.</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>US 2009-0254813 A1 (KOBAYASHI YUJI) 08 October 2009</td>
<td>1-15</td>
</tr>
<tr>
<td></td>
<td>See abstract and claims.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See abstract and claims.</td>
<td></td>
</tr>
</tbody>
</table>

Further documents are listed in the continuation of Box C. See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

'T' later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search
19 JULY 2011 (19.07.2011)

Date of mailing of the international search report
20 JULY 2011 (20.07.2011)

Name and mailing address of the ISA/KR
Korean Intellectual Property Office Government Complex-Daejeon, 189 Cheongsa-ro, Seo-gu, Daejeon 302-701, Republic of Korea
Facsimile No. 82-42-472-7140

Authorized officer
LEE, Jong Ick
Telephone No. 82-42-481-8373

Form PCT/ISA/210 (second sheet) (My 2009)
## INTERNATIONAL SEARCH REPORT

**Information on patent family members**

<table>
<thead>
<tr>
<th>Patent document cited in search report</th>
<th>Publication date</th>
<th>Patent family member(s)</th>
<th>Publication date</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>US 724631 1 B2</td>
<td>17. 07.2007</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2010-0863 15 A</td>
<td>15. 04.2010</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CN 101283348 A</td>
<td>08. 10.2008</td>
</tr>
<tr>
<td></td>
<td></td>
<td>JP 2009-510650 A</td>
<td>12. 03.2009</td>
</tr>
</tbody>
</table>

Form PCT/ISA/210 (patent family annex) (July 2009)