Visualización web interactiva de la demanda eléctrica en el Hospital de León.

Por

DIEGO GARCÍA PÉREZ

Departamento de Ingeniería Eléctrica, Electrónica, de Computadores y Sistemas
UNIVERSIDAD DE OVIEDO

Un TFM presentado a la Universidad de Oviedo en conformidad con los requisitos del MÁSTER EN INGENIERÍA DE AUTOMATIZACIÓN E INFORMÁTICA INDUSTRIAL.

FEBRERO 2017

Resumen

Los grandes edificios, como el Hospital de León, son importantes generadores de información, que en muchas ocasiones no es utilizada como herramienta para profundizar en el conocimiento de los mismos. En este trabajo se presenta una forma de explotar esta información para la mejora de la eficiencia energética. Si hablamos de eficiencia energética indudablemente uno de los campos en auge es el análisis y la representación inteligente de los datos de consumos eléctricos. Este trabajo se centra en estos dos conceptos, intentando fusionarlos en una aplicación web que proporcione al usuario una herramienta interactiva para conseguir una percepción intuitiva de la demanda energética del Hospital de León.

Una forma intuitiva, para el ser humano, de comprender cualquier concepto u objeto es desglosarlo en partes identificables a través de las cuales, componer mediante simples agregaciones conceptuales el elemento original. Si se traslada este análisis intuitivo al campo de la demanda eléctrica, consistiría en desagregar el consumo total del Hospital en patrones ocultos identificables, que sumados, resultan la potencia total consumida. En este trabajo se expone un mecanismo para conseguir esta representación basada en las partes de un todo, centrándose no solo en los resultados numéricos, sino en idear mecanismos para la codificación visual de esta representación, con el fin de crear una sinergia entre el análisis y la visualización.

Agradecimientos

Me gustaría agradecer a todas las personas que me han ayudado en la realización de este trabajo de investigación, en primer lugar a Ignacio Díaz Blanco quien me propuso el trabajo y quien ha aportado muchas de las ideas aquí desarrolladas, a Daniel Pérez López compañero de laboratorio por resolverme con gran paciencia las incontables dudas que le he planteado y por último a todo el personal del grupo SUPPRESS de la Universidad de León, el cual sin sus datos y su conocimiento sobre el Hospital de León este trabajo no habría sido posible.

Declaración del autor

Declaro que este trabajo ha sido realizado en acuerdo a los requerimientos del Máster de Ingeniería de la Automatización e Informática Industrial de la Escuela Politécnica de Ingeniería de Gijón. Excepto donde sea indicado por referencias específicas en el texto, el trabajo aquí presentado es el propio del candidato. Todos los puntos de vista aquí expresados son los del autor.

FIRMADO: ..... FECHA: .....

TABLA DE CONTENIDOS

ÍNDICE DE CUADROS

CUADRO **Página**

ÍNDICE DE FIGURAS

FIGURA **Página**

CAPÍTULO 1

INTRODUCCIÓN

Dentro de las problemáticas globales se encuentra el aumento de las emisiones de CO_2 atribuido, en parte, por el excesivo gasto energético. Dentro de este gasto, se pueden enmarcar los ámbitos residencial e industrial como dos de los sectores con mayor aportación en el consumo total. Debido a ello, se está despertando un gran interés en el uso de los recursos tecnológicos disponibles para la mejora de la *eficiencia energética* en procesos industriales y en grandes edificios.

Una de las lineas de investigación de actualidad en ciencia y tecnología, es el análisis y visualización de grandes cantidades de datos para extraer conocimiento de los mismos. Parece oportuno, por tanto, plantear un estudio de *eficiencia energética* apoyado en esta idea. En este capítulo, se justifica detalladamente la decisión de desarrollar un trabajo en el marco de *eficiencia energética* en un gran edificio, como es el Hospital Universitario de la ciudad de León, basándose en el uso de herramientas de gestión, análisis y visualización de datos. Se expondrá aquí el contexto del proyecto, el planteamiento del problema y los objetivos del trabajo.

1.1 Contexto del proyecto

En las últimas décadas, debido al aumento de la población mundial y el crecimiento económico e industrial de países en vías de desarrollo [1], el consumo energético global ha aumentado considerablemente figura 1.1. Este crecimiento trae consigo problemas de generación, agotamiento de los recursos naturales y un gran impacto medioambiental. Dentro de las múltiples consecuencias climáticas, se presta especial atención al *cambio climático* y en concreto a la subida de la tempe-

[1] Regiones como: El sudeste asiático, África o Sudamérica.

ratura del global. Las previsiones indican que la temperatura del planeta seguirá ascendiendo si no limitamos la emisión de CO_2 a la atmósfera, y por tanto, las consecuencias se agravarán desencadenando daños ambientales irreversibles a nivel mundial.

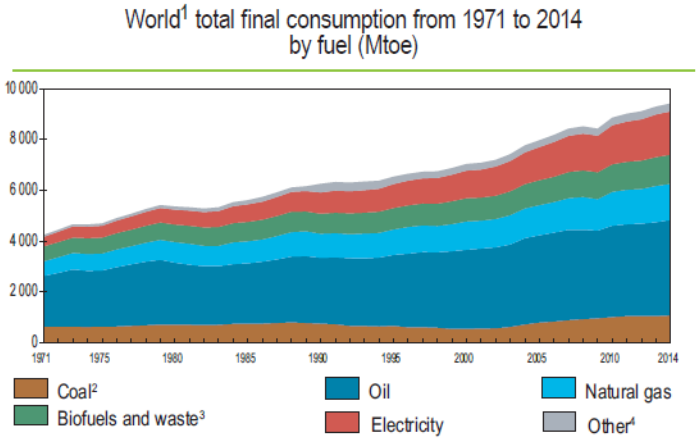

FIGURA 1.1. Consumo energético desagregado por recursos energéticos.(Fuente: International Energy Agency)

Recientes estudios [34, 36] muestran una estrecha relación entre el *producto interior bruto* (PIB) y la emisión de *gases de efecto invernadero*. Esta relación entre nivel económico y cantidad de emisiones, hace aún más justificable que en una sociedad moderna, el crecimiento económico debe ir a la par que el avance en medios para la reducción de emisiones, como puede ser la investigación en herramientas de *eficiencia energética*.

Según datos procedentes de la Unión Europea (Eurostat) [10], el consumo energético europeo descompuesto por sectores presenta la forma de la figura 1.2. Estos datos muestran que el consumo residencial supone un 24 % del consumo total en los países de la unión, siendo uno de los sectores con mayor demanda energética junto con el industrial y el sector transportes. Por tanto, está totalmente justificada cualquier investigación en la mejora de la eficiencia energética dentro de este sector.

Una vez centrada la problemática del trabajo en la eficiencia dentro del sector residencial, parece una buena opción el estudio de recintos públicos. Este tipo de edificios normalmente son grandes consumidores y su gestión está en manos de los gobiernos estatales o autonómicos, principales interesados en la eficiencia energética de los mismos, ya sea por una motivación puramente económica o por conciencia en la reducción de emisiones de CO_2. Estudios realizados por la *United States enviromental protection Agency* [9] (figura 1.3) muestran cómo es el consumo por unidad de superficie en distintos tipos de edificios. Se puede observar cómo un hospital posee una demanda energética superior a la media debido, en parte, a que suelen ser edificios de gran superficie, muy concurridos y que contienen sistemas de gran potencia muy dispares en

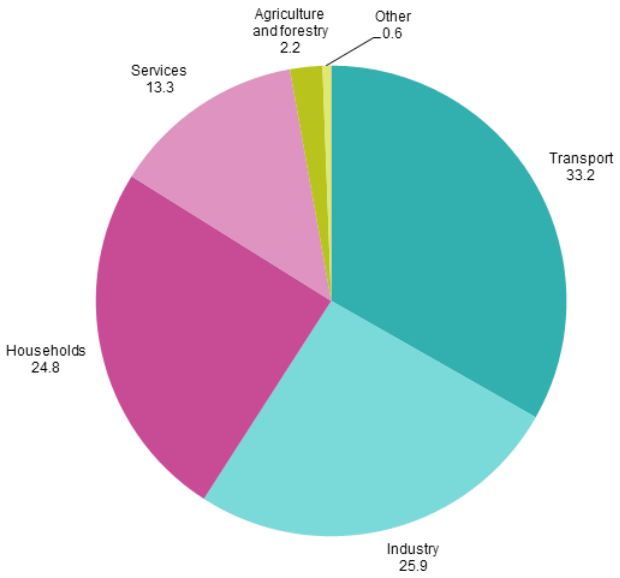

FIGURA 1.2. Consumo energético del año 2014 en Europa, desagregado sectores. (Fuente: Comisión Europea-Eurostat)

funcionalidad y en horarios de funcionamiento. Un ejemplo de este tipo de edificios es el hospital de la ciudad de León, este complejo está compuesto por una serie de edificios que desempeñan tareas muy dispares entre sí y que contienen sistemas de alta demanda energética.

Conocer aspectos como: la magnitud de los picos de consumo máximo y cuando se producen; la estructuración de la demanda eléctrica con respecto a ciclos temporales (horas, días, meses); o cómo se relacionan entre sí los consumos de distintas áreas del complejo, puede ayudar en la mejora de la eficiencia energética. En esta idea se centrará el trabajo, encontrar y aplicar herramientas que permitan extraer este conocimiento y presentárselo al usuario final. Estas herramientas deben facilitar el cambio en las estrategias de consumo, ayudar en el ajuste de tarifas o mejorar el rendimiento del sistema.

1.2 Planteamiento del problema

Recientemente, con la mejora de las tecnologías de medición y el abaratamiento de los costes en instrumentación, está aumentando la información obtenida de todo tipo de sistemas. Dicha información tratada con las herramientas adecuadas puede ayudar a profundizar en el conocimiento del sistema medido y a mejorar ciertas características del mismo. Este razonamiento encaja en el marco de la *eficiencia energética* en grandes edificios, es decir, con el análisis de mediciones

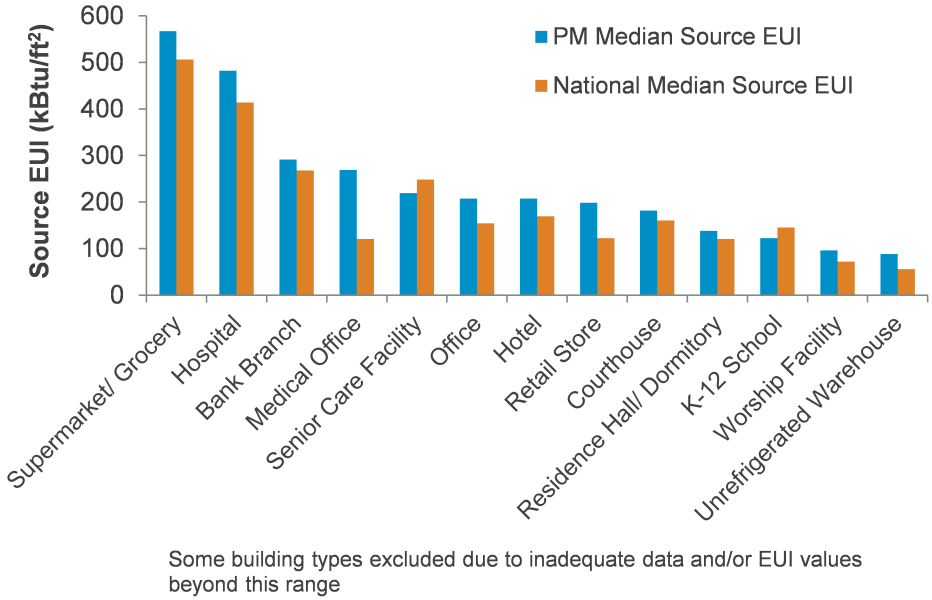

FIGURA 1.3. Consumo energético por tipo de edificio en EEUU según el programa "Energy Star". (Fuente: EPA-Energy Star)

eléctricas podemos encontrar aspectos desconocidos del edificio, que nos permitan ajustar las estrategias de consumo afectando lo más mínimo a su rendimiento.

Centrándonos ya en el Hospital de León, uno de los factores más influyentes dentro de la supervisión de la demanda eléctrica, y que es susceptible de estudio por el enorme interés que despierta entre el personal propio del hospital, es el consumo de los sistemas de confort térmico. Estos sistemas, según estudios del estado del arte [47], suponen entre 40-50% del consumo total en grandes edificios. En el caso particular del Hospital de León se cuenta con dos sistemas de confort térmico, uno de ellos es el sistema de calefacciones del que no se dispone de información, y el otro es el sistema de refrigeración. Este último, está compuesto por siete máquinas refrigeradoras con una potencia total de *1100 Kw* y que suministran agua fría a las *unidades de tratamiento de aire* (UTA) mediante un anillo de agua a baja temperatura. Las UTA son las encargadas de la refrigeración del hospital, y en especial de los aparatos de diagnóstico. Este sistema de refrigeración supone, como se ha mencionado, un alto porcentaje del consumo total del hospital, por lo que se debe prestar especial interés, dentro de nuestro análisis, a la búsqueda de patrones que reflejen el comportamiento de este sistema.

Para lograr aplicar técnicas de análisis sobre medidas eléctricas, es obvio que se debe obtener un conjunto de datos con dichas medidas. Por tanto, un primer planteamiento es indagar en la forma de obtener estas medidas procedentes de los distintos puntos de medida. Extraer únicamente las variables eléctricas, sin un mecanismo de acceso a ellas rápido e intuitivo, no

tiene ningún valor. Por este motivo, se debe tener en mente no solo extraer la información, sino organizarla y crear el software de acceso a ella.

Con un acceso ágil a los datos y teniendo en mente el análisis de las mediciones eléctricas para el estudio del comportamiento de la demanda eléctrica del hospital, el presente trabajo tiene como eje troncal, el uso de técnicas de *análisis inteligente de datos*(intelligent data analysis, IDA) para la búsqueda de patrones temporales de consumos. Dentro de la literatura referente al análisis inteligente de consumos eléctricos, existe una temática con creciente interés, denominada *energy disaggregation* o *Non Intrusive Load Monitoring* (NILM). Este tipo de técnicas consisten en descomponer un consumo global en suma de consumos desagregados, los cuales se corresponden a los diferentes elementos conectados a la red aguas abajo del punto de medida, utilizando algoritmos y métodos estadísticos. Teniendo esto en cuenta, se propone aplicar de este tipo algoritmos en los datos del hospital, con el fin de hallar patrones ocultos dentro de los datos de consumo. Las técnicas NILM pueden ser implementadas en base a diferentes algoritmos IDA, pero muy pocas aplicaciones se basan en el uso de técnicas *non-negative matrix factorization* (NMF). Por ello, plantear una investigación en técnicas NILM basadas en NMF, aporta originalidad y cierta relevancia al trabajo desarrollado.

Los resultados obtenidos por las técnicas de análisis propuestas, generalmente serán obtenidos en estructuras de datos codificados de forma numérica. Si se presentaran los resultados en este formato al usuario, nuestro análisis serviría de bien poco, ya que para el usuario sería difícil percibir los patrones obtenidos. Por tanto, se debe idear un mecanismo de presentación de los resultados eficiente. En el presente trabajo se propone el uso de la *percepción visual* del usuario como un poderoso instrumento a la hora de transmitir información procedente de los datos, incluso una apropiada presentación visual de la información, sin ningún análisis de relevancia, se podrían obtener conclusiones acerca del sistema. Estas técnicas de exploración son idóneas para extraer cierto conocimiento preliminar sobre el problema, pero lo verdaderamente interesante y en lo que ahonda este trabajo, es un desarrollo conjunto de técnicas IDA y herramientas de *visualización interactiva* para conseguir aplicaciones de *analítica visual* que aúnen estos dos campos.

1.3 Objetivos

En base a los planteamientos anteriores, podemos definir unos objetivos claros y razonables para este trabajo:

1. En primer lugar, realizar una capa de acceso a los datos, recogiendo y unificando todas las fuentes disponibles. Se pretende que estén recopilados en un mismo archivo, preferiblemente en una base de datos, de fácil acceso desde entornos de desarrollo típicos en análisis de datos.

El objetivo principal en esta fase es crear un banco de datos que se pueda usar no solo en este proyecto sino para análisis posteriores o paralelos a este.

2. Aplicar sobre las mediciones recopiladas, herramientas NILM no supervisadas basadas en NMF, explorar sus posibilidades, analizar sus resultados y la forma de interpretarlos. En esta etapa del trabajo, se debe hacer una investigación dentro de la literatura para elegir que implementación NMF brinda una mayor interpretabilidad a la hora de desarrollar aplicaciones visuales. Por último tras haber obtenido ciertos resultados plausibles, se plantea como objetivo buscar conexiones entre los patrones encontrados y eventos conocidos dentro del hospital, con la finalidad de verificar la validez de los resultados.

3. Realizar una aplicación visual que muestre los resultados obtenidos y se base en los principios fundamentales de la percepción visual del ser humano. Se pretende que sea operativa en distintos dispositivos, ágil, e interactiva. Hoy en día una de opciones mas frecuentes para conseguir este tipo de aplicaciones es un entorno *web*, basado en *html* y *Javascript*.

4. Por último en el desarrollo de la aplicación, se fija como objetivo seguir una metodología *Model-View-Controller* [2]. Utilizando MVC se pretende integrar en una misma aplicación web, la ejecución de los algoritmos de análisis, y la aplicación interactiva que muestra los resultados. De esta forma se agiliza el proceso de pruebas y permite al usurario modificar parámetros del análisis que alteran la respuesta visual de la aplicación,como se muestra de forma esquemática en la figura 1.4. Esto, permite asimilar todas las posibilidades del análisis con más profundidad que mediante un desarrollo en el que la algoritmia y las visualizaciones no estén integradas.

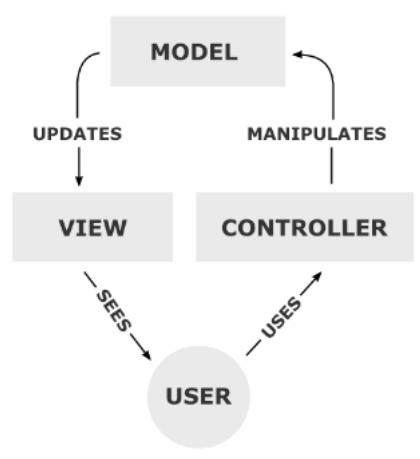

FIGURA 1.4. Esquema patrón de diseño MVC. (Fuente: Wikipedia)

[2]Patrón de diseño ampliamente extendido en ciencias de la computación

CAPÍTULO 2

MÉTODOS Y TÉCNICAS

De entre las diferentes metodologías con las que abordar un trabajo de análisis de datos como el presente, una de las más extendidas es el estándar *CRISP-DM (Cross Industry Standard Process for Data Mining)* [52]. Como se ilustra en la figura 2.1, el estándar desgrana un proyecto de análisis de datos en una serie de tareas. Inspirándose en ellas, el presente trabajo se ha dividido en las siguientes fases: extracción de los datos, preparación de los datos, análisis y visualización de los resultados. En las diferentes secciones de este capítulo se indicará las metodologías y herramientas software necesarias para realizar cada una de estas tareas propuestas.

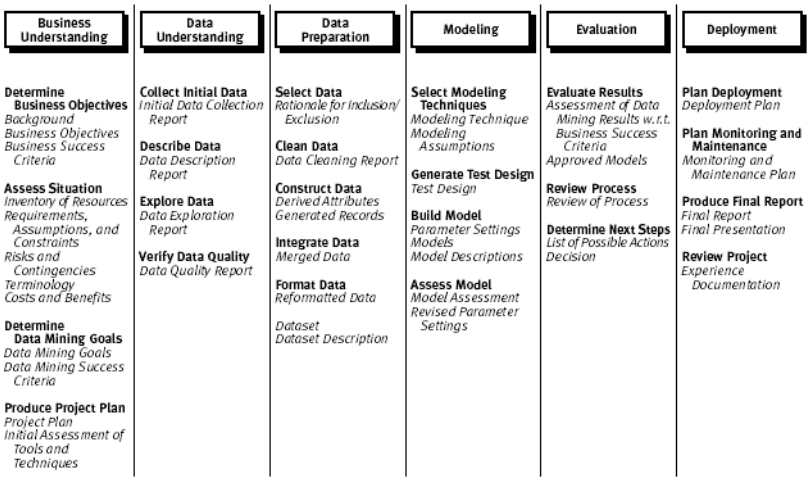

FIGURA 2.1. Tareas en CRISP-DM.(Fuente: [52])

2.1 Extracción de los datos

Teniendo como principal objetivo la unificación de toda la información recogida por los medidores inteligentes del hospital, es necesario diseñar y desarrollar una herramienta software que recopile, estructure y disponga, de manera ágil para usuario, toda la información medida en el hospital. Los aspectos de diseño y las herramientas software, utilizadas para la implementación de dicha capa, serán explicadas en lo que queda de sección.

2.1.1 Hardware y software encargado de las mediciones eléctricas

En la última década, con el objeto de una supervisión del suministro energético, el personal técnico del Hospital de León en colaboración con el grupo de investigación *SUPPRESS* de la Universidad de León [23], ha venido desarrollando una arquitectura para la medición de una gran variedad de variables eléctricas. Para entender la necesidad de una capa de gestión y la manera en la que se ha desarrollado, en los siguientes párrafos se resumirán aspectos fundamentales, tanto de software como de hardware, de la arquitectura que se encarga de recoger las distintas mediciones eléctricas. Entre estos aspectos, se incluyen el tipo de medidores utilizados, la localización en la red de los puntos de medida y el software de recogida de los datos.

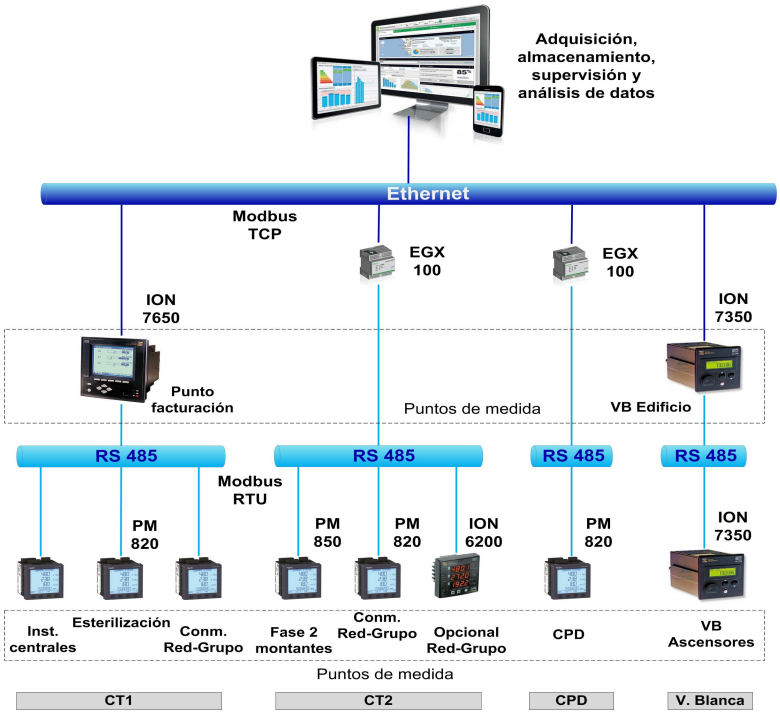

FIGURA 2.2. Arquitectura de red para supervisión de la demanda eléctrica en el Hospital de León.(Fuente: grupo SUPPRESS). Los rectángulos discontinuos engloban los diferentes puntos de medición eléctrica, mientras que los rectángulos rellenos indican donde se ubican los medidores

Dentro de la red de suministro energético del Hospital de León, se encuentran conectados diferentes equipos de medida de la familia Scheneider Electric, como son: la familia de medidores PM800, ION6200, ION7650 y ION7350. Todos ellos, cuentan con capacidades de medida de gran rendimiento en tensiones, intensidades, potencias así como en mediciones de calidad de onda. Una descripción más detallada de los medidores utilizados se puede encontrar en la web de Schneider Electric [20].

Ubicación	**Punto de medida**	**Medidor**	**Concentrador**
Centro de Transformación 1	Punto de facturación	ION7650	ION7650
	Inst. Centrales	PM820	ION7650
	Esterilización	PM820	ION7650
	Conm. Red-Grupo	PM820	ION7650
Centro de Transformación 2	Fase 2 montantes	PM850	EGX100
	Conm. Red-Grupo	PM820	EGX100
	Opcional Red-Grupo	ION6200	EGX100
Centro Procesado de Datos	CPD	PM820	EGX100
Edif.Virgen Blanca (VB)	VB. Edificio	ION7350	ION7350
	VB. Ascensores	ION7350	ION7350

Cuadro 2.1: Puntos de medida eléctrica Hospital de León

La figura 2.2 ilustra que, para la supervisión de la demanda de eléctrica, el hospital cuenta con una arquitectura de red en la que se incluyen los medidores Schneider Electric, pasarelas y distintos protocolos de comunicación industrial. Como se puede apreciar, varios medidores PM800 están conectados mediante *Modbus RTU* (capa física RS-485,[15]) a medidores ION. Los medidores ION concentran la información recibida por los medidores PM, y encapsulan Modbus RTU en Modbus TCP para reenviar la información concentrada de los medidores a un servidor aguas arriba. No todos los medidores PM se encuentran conectados a medidores ION, algunos de ellos se encuentran conectados a dos pasarelas EGX Modbus RTU/Modbus TCP, las cuales realizan el encapsulado en Modbus TCP de la información medida. De esta manera, toda la información es recibida por un servidor, que mediante un software propietario de Schneider almacena la información medida.

Los rectángulos discontinuos en la figura 2.2 engloban los distintos puntos de medida, todos estos puntos de medida se recogen de forma ordenada en el cuadro 2.1. Toda la información medida en estos nodos es recogida por un servidor alojado en el propio hospital. Esta máquina recopila la información a través del software propietario de Schneider *Power Monitoring Expert* (PME). *"Power Monotoring Expert es un paquete de supervisión potente para aplicaciones de gestión de la demanda eléctrica, recoge y organiza los datos resenteándolos mediante una interfaz web intuitiva"* [17]. Para la creación del sistema de supervisión, PME recoge todas las medidas dentro de una base de datos *Microsoft SQL Server* (MS-SQL). Esta base contiene una gran

cantidad de tablas, la mayoría de ellas innecesarias para nuestro propósito, organizadas de forma poco intuitiva. Esta estructura de tablas provoca que las consultas sean complejas y poco ágiles. Además, por razones de espacio de almacenamiento en el servidor, el personal del hospital cada 3 meses vuelca las bases a servidores de la Universidad de León, reiniciando toda la base y provocando que los identificadores de los medidores cambien, provocando la pérdida de trazabilidad de los datos cada 3 meses. Por todas estas razones, es necesario crear una aplicación de gestión con capacidad para leer todas las bases trimestrales y para generar una nueva base con una estructura de tablas sencilla, funcional y con identificadores invariantes, que brinde al usuario un acceso rápido a los datos sin necesidad de consultas complejas. Para desarrollar una aplicación de estas características, se debe decidir con qué herramientas se va a afrontar las problemáticas planteadas. En la siguiente sección, se argumenta por qué se ha elegido la combinación de Python, SQLAlquemy y Pandas como pilares para el desarrollo de esta aplicación.

2.1.2 Gestión bases de datos: Python, SQLAlquemy y Pandas

Dentro de los lenguajes actuales, Python cuenta con una gran aceptación dentro de la comunidad tecnológica [2], especialmente en campos como el análisis y visualización de datos [3]. Esta popularidad es debida, entre otras muchas razones, a que la filosofía del lenguaje desde su creación, en la década de los ochenta, es proporcionar al usuario una sintaxis que favorezca la legibilidad del código. Por ello, está diseñado para ser un lenguaje interpretado, multiparadigma, multiplataforma, de tipado dinámico y que proporciona una gran flexibilidad al programador cómo se explica en su web [19]. Además de todas estas razones, a lo largo de este capítulo, el lector obtendrá diversos argumentos que justifican el uso de Python en las distintas fases del trabajo.

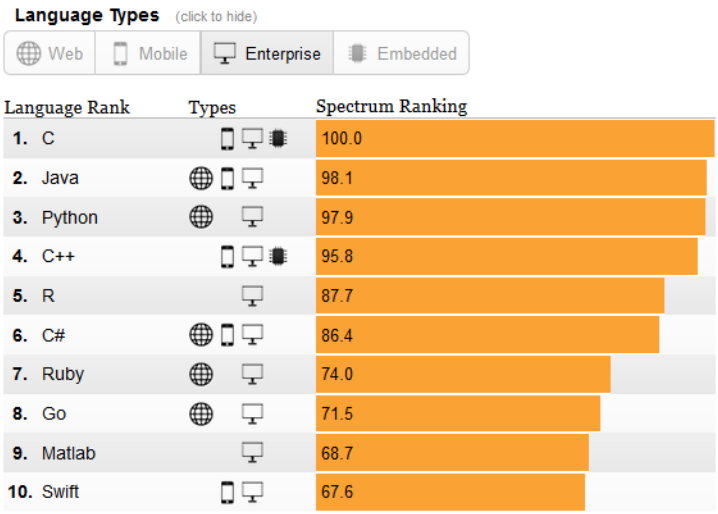

FIGURA 2.3. Lista lenguajes más utilizados 2016.(Fuente: IEEE Spectrum)

Para la gestión de la información almacenada en las bases de datos del hospital, el uso de Python está ligado a la existencia del paquete software *SQLAlquemy* [22], el cual provee de una capa de abstracción entre nuestra aplicación de gestión y el motor de base de datos basado en SQL. Este tipo de módulos, permite el manejo de bases de datos relacionales, como *Microsoft SQL Server* (MS-SQL), dentro del paradigma de programación orientada a objetos. De esta forma, podemos gestionar bases de datos mediante sencillas llamadas a métodos dentro de Python, sin necesidad de un extenso conocimiento sobre motores de bases de datos y lenguajes SQL. En la figura 2.4 se muestra como SQLAlquemy se compone por un núcleo, el cual se apoya en una DBAPI, y puede conectarse a un motor de bases de datos para realizar consultas abstractas a la base conectada. Dicha conexión se realiza mediante la instancia de un objeto propio del módulo SQLAlquemy denominado *engine*, el cual proporciona potentes métodos para realizar las consultas. Dentro de los motores de base de datos que se pueden conectar el núcleo de SQLAlchemy, se encuentra MS-SQL lo cual es idóneo, ya que los medidores repartidos hospital almacenan los datos bajo la gestión de MS-SQL.

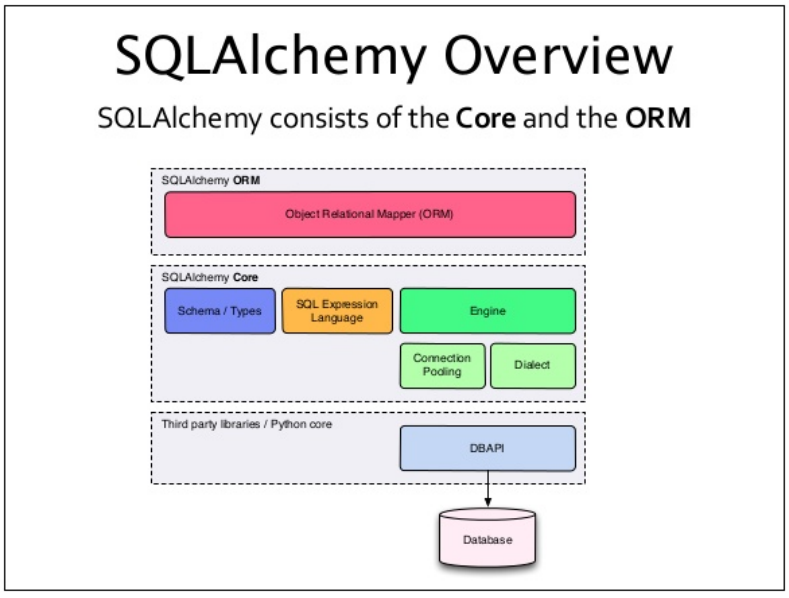

FIGURA 2.4. Capas que conforman SQLAlquemy.(Fuente: IEEE Spectrum)

En esta fase del trabajo será necesario el manejo de grades cantidades de datos multivariables asociados a índices temporales. Para este tipo de tareas, Python cuenta con otro módulo bastante conocido en el campo del análisis de datos, como es *Pandas*. *"Pandas es una librería de código abierto, que proporciona herramientas potentes y de fácil uso para el tratamiento de estructuras de datos y análisis de datos"* [16]. Para el manejo de datos multivariable, pandas permite instar estructuras de datos propias, las cuales contienen una serie de métodos que nos permiten aplicar

un amplio espectro de operaciones sobre los datos. La estructura de datos más común en Pandas es el *DataFrame*, la cual es una estructura de datos bidimensional y consta de un índice que identifica cada fila y de columnas que representan los diferentes atributos.

Un DataFrame dispone los datos de forma similar a una tabla en una de las bases relacionales más comunes. Por ello, parece razonable utilizar este tipo de estructuras para recoger y tratar los datos procedentes de las bases MS-SQL de los medidores. Gracias a la flexibilidad que nos aporta la unión entre Pandas y SQLAlquemy, podemos obtener los datos de consultas SQL en formato DataFrame de forma directa. Pandas utiliza como parámetro el objeto *engine* de SQLAlquemy y nos brinda una serie de métodos para hacer consultas SQL. De esta forma se hace posible acceder y manejar los datos de forma transparente al usuario, y desarrollar en un mismo entorno, herramientas que combinen la extracción de la información con la potencia que ofrece Python en análisis y visualización de datos.

2.1.3 Aplicación para la extracción de datos

A la hora de explicar la capa de software desarrollada para la unificación de las bases trimestrales, se hace hincapié en: cual es la información relevante; cómo se ha extraído, procesado y almacenado en la nueva base; y en cual es la nueva estructura de tablas elegida.

Tras trabajos previos de exploración por parte del grupo *SUPPRESS*, la ***información relevante*** en las bases trimestrales, se ha localizado en tres tablas:

- ***Source***: tabla que recoge los medidores registrados en la arquitectura de red. Dentro de sus muchos campos, destacan los atributos:
  - *Name*: nombre de los medidores registrados en la red.
  - *ID*: ID único asociado a cada medidor.
- ***Channel***: esta tabla recoge la relación entre variables y medidores. Destacan los campos:
  - *QuantityID*: ID de las variables.
  - *Label*: nombre de las variables.
  - *RecorderID*: ID de medidor, acorde a la tabla Source.

  Existen varios registros con QuantityID idénticos, pero con distinto RecorderID.
- ***Datalog***: En esta tabla se guarda los datos en crudo de todas las mediciones. Tiene como atributos:
  - *SourceID*: ID de medidor, acorde a la tabla Source.

- *QuantityID*: ID de variables, acorde con la tabla Channel.
- *Value*: valor de la magnitud identificada con los ID de variable y medidor.
- *Timestamp*: instante temporal en el que se produce la medida, con resolución de milisegundos.

En los registros almacenados, una combinación de SourceID-QuantityID-TimeStamp es única, es decir esos tres campos formarían la clave de la tabla. Las mediciones se almacenan con periodo de muestreo de minuto y cuentan con una gran cantidad de variables y medidores; provocando que esta tabla contenga millones de registros.

El grupo *SUPPRESS*, realiza trimestralmente el volcado de las bases junto con la restauración del servidor PME. De la base recién extraída del PME, solo son almacenadas, en servidores propios de SUPPESS, las tres tablas indicadas. Los tríos de tablas extraídas trimestralmente son guardadas con un identificador del trimestre en el que han sido extraídas, en una misma base denominada *ElectricidadHospital*.

Una vez se conoce la información relevante, debemos definir una metodología eficiente para ***extraer*** las tablas indicadas mediante pandas y SQLAlquemy. Con el fin de un código encapsulado, se ha desarrollado una clase *Lector*, encargada de extraer las tablas mencionadas. Esta clase almacena como atributo *driver* que no es más que un objeto *engine* de SQLAlquemy, con el que es capaz de realizar las consultas a las bases MS-SQL. En el siguiente código se presenta una porción del archivo *Lector.py*, con el constructor de la clase Lector:

```
import pandas.io.sql as psql
import pandas as pd
from sqlalchemy import create_engine

class Lector:
        def __init__(self,ODBCName = 'ElectricidadHospital',User = '
            Electricidad',Psw = 'maquetas'):

                self.ODBCName = ODBCName
                self.User = User
                self.Psw = Psw
                self.driver = create_engine('mssql+pyodbc://' + self.User + ':'
                     + self.Psw + '@' + self.ODBCName)
```

Se puede observar cómo mediante la instrucción `create_engine`, procedente de SQLAlquemy, se crea un objeto *engine*, dentro de los atributos de la clase lector. En el siguiente ejemplo se muestra una instancia del objeto lector:

```
from Lector import *
```

```
L = Lector(ODBCName = 'ElectricidadHospital',User = 'Electricidad',Psw = '
    maquetas')
```

A través de una instancia del objeto `Lector`, como la anterior, podemos acceder a sus métodos, los cuales han sido diseñados para realizar las consultas más comunes al tipo de tablas descritas, en las que principalmente se almacenan identificadores, valores flotantes, fechas y horas. Teniendo en cuenta este tipo de datos se han desarrollado dos métodos:

- ***Consulta***: método diseñado para consultas concretas, es decir para extraer fragmentos de la tabla indicada. Se pueden filtrar los datos devueltos por: columnas, sentencias condicionales y mediante fecha y hora del día. Los datos serán devueltos en formato DataFrame, propio de Pandas. El siguiente fragmento de código es un ejemplo para una consulta de las columnas Value y TimeStamp de la tabla Datalog, donde SourceID es igual a 4 y de TimeStamp esta comprendido entre febrero de 2012 y febrero de 2013:

```
L.consulta(Columnas=['Value','TimeStamp'], Tabla='DataLog',
    condiciones = [('ID','4')], fechas = ('TimeStamp',pd.
    to_datetime('2012-02-03'),pd.to_datetime('2013-02-03')))
```

- ***Tabla***: es el método más utilizado en nuestro desarrollo y tiene la capacidad de leer una tabla entera. Para la lectura de la tabla DataLog, al contener millones de registros, se ha tenido que añadir la funcionalidad de la lectura por lotes mediante un iterator de Python, el cual está incluido en las *SQL Queries* de Pandas [13]. De la misma forma que `consulta`, los datos son devueltos en un DataFrame. En el siguiente ejemplo se muestra como se realiza la lectura de la tabla DataLog:

```
rDataLog = L.tabla(Tabla='DataLog', chunksize = 10000)
```

Una vez desarrollado un mecanismo para la lectura de las bases trimestrales, el siguiente paso en la construcción de nuestra aplicación de gestión es un diseño para la ***estructura de tablas*** de la nueva base. Siguiendo las reglas de normalización de bases de datos [26, 32], se ha optado por el diseño mostrado en la figura 2.5. En la figura se muestran las tablas y campos que se han elegido y las relaciones entre las tablas. La distribución es parecida a la propia de Schneider, con la diferencia de que se añade la tabla *MedVarPME*, para solventar la relación "n a n" entre las tablas de medidores y variables. Esto es así, ya que un medidor contiene varias variables y una variable es medida por varios medidores. Solo se puede implementar esta relación, mediante el uso de una tabla intermedia que asocie un *IDMV* único a pares *IDMed-IDVar*. Este ID es la clave principal de la tabla datos, la cual está relacionada "1 a n" con *MedVarPME*.

A la hora de crear esta estructura, una vez más, se hace uso de una clase en Python que tiene como atributo un objeto *engine* de SQLAlquemy. Esta clase, denominada `Tablas`, se encarga

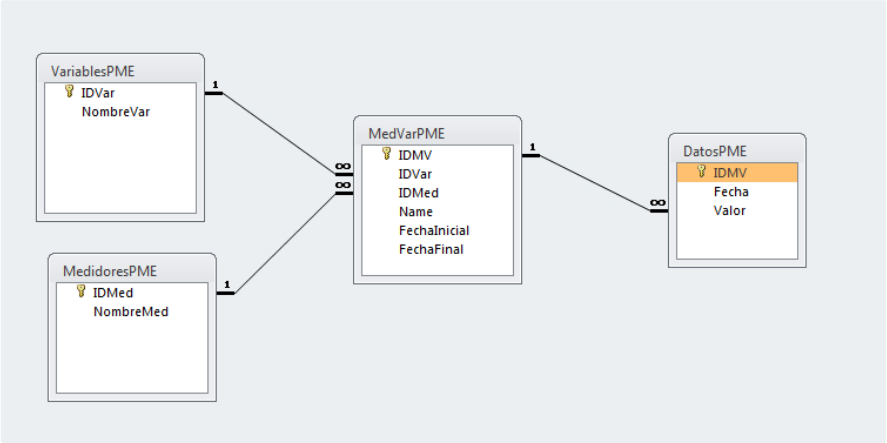

FIGURA 2.5. Diseño tablas propuesto para la aplicación de gestión

de generar la estructura planteada en una base MS-SQL [1]. Para ello, cuenta con el método `Tablas.crear()`, que haciendo uso de funciones propias de SQLAlquemy, crea las tablas con los campos indicados en la figura 2.5.

Tras haber creado las tablas y tener en memoria las tablas trimestrales, se procede a ***procesar*** la información y generar los nuevos datos a almacenar. Como se introdujo en la sección 2.1.1, con el volcado trimestral, los ID de variables y medidores no permanecen invariantes pero sí lo hacen los nombres de las variables y medidores. Apoyandose en eso, se consigue generar ID propios e invariantes, comparando de los nombres de los medidores y variables de las tablas entrantes con los nombres ya almacenados en la nueva base, si los hubiere. En el caso de la primera carga de datos se generarán los ID iniciales y se guardarían por primera vez los nombres de las variables y medidores; esta situación también puede darse cuando se inicia la medición de nuevas variables. En este caso se actuará con el mismo criterio, se guardarán sus nombres y se les asignará un medidor.

Una vez están correctamente actualizadas las tablas *MedidoresPME*, *VariablesPME* y *MedVarPME* con los ID propios, se apilan los datos extraídos de la tabla *Datalog* en la nueva tabla *DatosPME*. Cabe resaltar que los atributos *FechaInicial* y *FechaFinal* de *MedVarPME*, son necesarios para evitar solapamientos en los datos apilados en *DatosPME*. En este punto, todos los datos procesados y listos para cargar en la base, se encuentran almacenados en RAM, con

[1] Se decidió realizar la nueva base en el mismo motor, ya que se disponía de las licencias pertinentes. Pero se podría implementar en cualquier otro motor soportado por SQLAlquemy.

formato DataFrame.

Para el *almacenado* de los Dataframe procesados se crea una vez más una clase, pero a diferencia de `Lector`, no usa como driver el objeto *engine* de SQLAlquemy, sino que se hace uso del módulo *pypyodbc* [18]. La razón por la que se ha elegido este módulo, radica en que permite hacer uso de la funcionalidad *BULK INSERT* de T-SQL [24], mientras que SQLAlquemy no. Para la carga, mediante *BULK INSERT*, los DataFrame procesados deben ser guardados como un archivo *.csv* mediante el método `to_csv` de Pandas.

2.2 Preparación de los datos

Concluida la fase de *extracción de datos* del hospital existe una etapa de preparación de los datos, previa al análisis y modelado de los mismos. Esta fase del trabajo pretende dar respuesta a las siguientes cuestiones: ¿De entre todas las medidas, cuáles son las más apropiadas para iniciar nuestro análisis ?¿Cuál será el formato elegido para datos de partida en el análisis?¿Existen mediciones ausentes o erróneas, y en caso afirmativo, como las tratamos? ¿Es suficiente con los datos medidos o necesitamos completar esta información?

2.2.1 Selección de la información a tratar

En primer lugar decir que el número total de variables eléctricas recogidas por todos los medidores asciende a 765 y la mayoría de ellas se han estado registrando durante varios años con periodo de muestreo igual a un minuto. Esto hace que tengamos una gran cantidad de datos, muchos más de los que podemos abarcar en un trabajo como el presente, por ello, debemos elegir, de entre todas las variables disponibles, las variables más adecuadas para nuestro análisis posterior. Siendo este un trabajo de análisis de datos basado en técnicas de desagregación de consumos (técnicas NILM), parece razonable extraer variables de potencia las cuales sean suma de grandes consumos dentro del hospital. Además de esto, las variables elegidas deben ser de interés para la supervisión de la demanda eléctrica en el hospital. Teniendo en cuenta estos dos aspectos, se ha decidido extraer únicamente las medidas de potencia total en el punto de facturación. Los kW medidos en el punto de facturación, recogen todos los consumos aguas abajo de la acometida general del hospital, como por ejemplo, los consumos de las siete máquinas del sistema de refrigeración. Este sistema, anteriormente mencionado en el capítulo 1, es de especial interés en nuestro trabajo.

Otro de los aspectos influyentes a la hora de elegir el conjunto de datos a extraer es qué periodo de tiempo elegir. Dos razonamientos justifican el periodo de datos a extraer: el primero de ellos, es que el conjunto de datos elegido debe ser aquel, que entre todos los datos disponibles, tenga menor cantidad de errores de medida o huecos; el segundo razonamiento, es la influencia del conjunto elegido en los resultados del análisis. Así, si se elige el periodo de un mes, tendremos resultados cuya interpretación está fuertemente ligada a los patrones semanales o diarios, mientras que si el

conjunto abarca un año, el análisis extraerá patrones mensuales o estacionales. En esta decisión se debe tener en cuenta, el enorme interés por el comportamiento del consumo de las máquinas enfriadoras, el cual, tiene fuerte relación con las estaciones del año. Es, por todo ello, conveniente extraer la potencia total en kW del hospital del año 2014, ya que es el año con menos datos perdidos, recoge el consumo del sistema de refrigeración y nos permite el estudio de patrones estacionales.

2.2.2 Pretratamiento mediante Pandas

Los datos serán extraídos de la tabla *DatosPME*, por medio de la clase `Lector`, y debido a que las consultas a bases *MS-SQL* suelen tener un tiempo de computo del orden varios minutos, se decide guardar el conjunto elegido en un archivo *KwTEneroDiciembre.csv*. De esta forma, tenemos un conjunto de datos, a los que podemos acceder de forma ágil y sin necesidad de conectarnos a la base de datos, listos para las pruebas en la etapa de pretratamiento y análisis.

La lectura de los datos guardados en el archivo *KwTEneroDiciembre.csv*, se lleva a cabo mediante Pandas de la siguiente forma:

```
import pandas as pd
KwT = pd.read_csv("./KwTEneroDiciembre.csv",index_col = "Fecha")
```

La función `Pandas.read_csv()` devuelve un DataFrame que por columnas tiene los campos de la tabla *DatosPME* y por índice de filas la fecha y hora del campo *Fecha*, como se indica en la función `read_csv` mediante el parámetro *index_col*. El índice indicado, es leído como un array de strings, pero nos interesa que sea un tipo de datos de fecha/hora. En el siguiente fragmento de código se muestra como pandas nos permite, mediante la función `pandas.to_datetime()`, convertir el índice a *datetime64*, un tipo de dato propio de Python para datos del tipo fecha/hora:

```
KwT.index = pd.to_datetime(KwT.index)
```

Como se ha mencionado con anterioridad, en ciertas ocasiones los datos de partida tienen huecos, estos se producen porque el servidor PME deja de recibir las medidas por fallos temporales en la red del hospital. Estos huecos provocan que en nuestro DataFrame no tenga un registro por cada minuto del año, es decir, de los 525600 minutos que tiene un año, el DataFrame obtenido contiene 482909 filas. Para completar la información que aporta el DataFrame, debemos generar registros para los minutos faltantes con valor *not a number* (NaN). Para ello, como se muestra en siguiente fragmento de código, se genera un índice para ese año, mediante las funciones propias de Pandas `Pandas.date_range` y `DataFrame.reindex()`.

```
from pandas.tseries.offsets import *

indexmin = pd.date_range(start=min(KwT.index), end = (max(KwT.index) +
    DateOffset(minutes=1)), freq='min')
```

```
KwT = KwT.reindex(index = indexmin,method = None)
```

Se generan un índice desde la fecha y hora mínima, hasta la máxima, y mediante la función `reindex` se ajustan los datos a este nuevo índice. El parámetro `method`, índica con qué método rellena los huecos al adecuar los datos al nuevo índice; eligiendo el valor *None* los rellenará con NaN.

Por último, en la etapa de preparación de los datos se le añade información adicional, en forma de nuevas columnas. Estos nuevos atributos son información temporal extraída del índice y cuantificada como enteros. El día del año o mes, la hora o minuto del día son ejemplos de las columnas adicionales. Esta tarea es sencilla gracias a los atributos propios del índice (index), como se muestra a continuación:

```
KwT['DayOfYear'] = KwT.index.dayofyear
KwT['Hour'] = KwT.index.hour
KwT['Time'] = KwT.index.time
KwT['Month'] = KwT.index.month
KwT['DaysInMonth'] = KwT.index.day
```

Usando estas columnas como nuevas características de los datos, nuestro análisis posterior de los datos puede ser dotado de nuevos enfoques temporales, permitiendo nuevos modos de análisis y reestructuraciones temporales de los datos, como se verá en la siguiente sección.

2.3 Análisis inteligente: Energy Disaggregation y Non-Negative Matrix Factorization

2.3.1 Concepto de desagregación de energía

Un enfoque interesante dentro del campo del análisis inteligente de la demanda eléctrica es la *desagregación energética* o *energy disaggregation* [40]. Esta disciplina descompone una medición agregada en varias componentes, revelando patrones característicos aguas abajo del punto de medida. Un ejemplo ilustrativo se da en análisis del consumo de una vivienda familiar, donde el objetivo de este tipo de técnicas es obtener los consumos de los diferentes aparatos conectados en el interior de la vivienda. En los últimos años, el interés en la desgregación de energía también conocida como *non-intrusive load monitoring* (NILM) ha crecido, ya que la percepción y el conocimiento del consumo se mejorado a través de una representación basada en las “partes de un todo". Además, este tipo de representación puede sugerir cambios en horarios, o detección de fallos en la red de suministro energético.

Las técnicas NILM se dividen en métodos *supervisados* o *no-supervisados*. Dentro de los métodos supervisados, la mayoría de aplicaciones en la literatura [37] están basadas en algoritmos

de reconocimiento de patrones y optimización, y su principal desventaja es que se necesita un conjunto de datos etiquetado. En muchos casos, obtener datos debidamente etiquetados puede incrementar los costes de implementación de los sistemas NILM, haciendo que los métodos NILM no-supervisados concentren mayor interés. Dentro de los métodos NILM no-supervisados, *blind source separation* (BSS) y *factorial hidden Markov models* (FHMM) como se explica en [54]. El presente trabajo se centrará en los métodos BSS en especial en *non-negative matrix factorization*(NMF). NMF,al igual que técnicas como *principal component analysis* o *independent component analysis* [39, 46], consite en una factorización matricial con ciertas restricciones. La principal restricción, en la factorización matricial llevada a cabo por algoritmos NMF, es la no negatividad en los elementos de la matrices resultantes y de entrada, mientras que las restricciones en las técnicas PCA e ICA son respectivamente, la ortogonalidad y la independencia estocástica.

Las medidas de demanda eléctricas son siempre positivas, por tanto NMF es una técnica que se puede aplicar en estos datos, y generar una descomposición con el fin de conseguir una representación basada en partes de un todo, como se muestra en [41]. La aplicación de estas técnicas, basadas en suma de elementos positivos, a la demanda eléctrica del Hospital de León, se pretende descubrir patrones ocultos en los consumos y relacionarlos con eventos característicos en la red, como por ejemplo, arranques de las máquinas refrigeradoras.

2.3.2 Algoritmo NMF

NMF es formulado [41, 45] como una factorización matricial aproximada de una matriz de entrada no-negativa expresada de la siguiente manera:

$$\mathbf{V} \approx \mathbf{WH} \tag{2.1}$$

Si definimos un vector de M dimensiones, $\mathbf{v}_i$, cuyos elementos son no-negativos y la matriz $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N] \in \mathbb{R}^{M \times N}$ como N observaciones de $\mathbf{v}_j, j = 1...N$. Considerando $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_L] \in \mathbb{R}_+^{M \times L}$ y $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N] \in \mathbb{R}_+^{L \times N}$, la ecuación 2.1 puede reformularse como:

$$\mathbf{v}_j = \sum_{\alpha=1}^{L} \mathbf{w}_\alpha h_{\alpha,j} \tag{2.2}$$

donde $\mathbf{v}_j$ es la j-ésima columna de $\mathbf{V}$ y $h_{\alpha,j}$ es la α-ésimo elemento de la j-ésima columna de $\mathbf{H}$. Por lo tanto, $\mathbf{v}_j$ es una combinación lineal de las columnas de $\mathbf{W}$. Los coeficientes de la combinación vienen impuestos por los elementos de las columnas de $\mathbf{H}$, así, todas las observaciones de entrada son suma ponderada de las bases no-negativas. Esta no-negatividad sugiere que las columnas de $\mathbf{W}$ son "las partes de un todo" como se explica en [42]. Estas columnas, en la aplicación concreta del algoritmo a consumos eléctricos, serán denominadas *consumos base* o simplemente

componentes. Por lo tanto, los elementos de las columnas de **H** contienen información sobre la influencia de los consumos base a lo largo de las N observaciones. La aplicación de NMF a los consumos eléctricos del Hospital de León, permite extraer un cierto número de consumos base que son una representación por partes de un consumo total, como es el punto de facturación. Estos consumos base revelan patrones latentes, los cuales pueden contener información útil para la mejora del conocimiento del sistema.

Una de las desventajas de este tipo de técnicas es determinar el número L de consumos base con el cual el algoritmo converge en una descomposición exacta $\mathbf{V} = \mathbf{WH}$. Estimar el número de componentes a descomponer es un problema *NP-completo* como se introduce en [50]. Debido a la dificultad de estimar L, se desestima el modelo exacto [50] y en el resto del desarrollo del trabajo se hará uso del modelo básico de la ecuación 2.1, al que se le añadirán más restricciones además de la no-negatividad.

Típicamente, **W** y **H** son estimadas mediante un problema optimización, cuya *función objetivo* es una medida de similitud entre **V** y **WH** [42] y que es resuelto por técnicas de gradiente descendente [43]. Este problema es no convexo con respecto a **W** y **H** al mismo tiempo; sin embargo, por separado es convexo con respecto, tanto a **W**, como a **H**. Como un problema de optimización no convexo basado en métodos de gradiente descendente, los consumos base obtenidos como resultado del modelo básico de NMF dependen de la inicialización de las matrices **W** y **H**. Para esta inicialización se considera el uso de *nonnegative double singular value decomposition* (*nndsv*) [33].

Una variante del método básico, introducida en [38], is NMF *disperso*, cuyos resultados revelan componentes mas interpretables, ya que solo unos pocos elementos de **W** y **H** son significativos y el resto son próximos a cero [2]. Aunque él módelo básico de NMF en problemas determinados problemas alcanza soluciones dispersas, las variantes de NMF disperso incluyen nuevas restricciones en la función objetivo, así se garantiza resultados dispersos.

Con este nuevo enfoque, la función objetivo sería de la siguiente forma:

$$\arg\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{V}-\mathbf{WH}\|_F^2 + \alpha\rho\|\mathbf{W}\|_1 + \alpha\rho\|\mathbf{H}\|_1 + \frac{\alpha(1-\rho)}{2}\|\mathbf{W}\|_F^2 + \frac{\alpha(1-\rho)}{2}\|\mathbf{H}\|_F^2 \tag{2.3}$$

donde $\|\mathbf{V}-\mathbf{WH}\|_F^2$ es una aproximación por mínimos cuadrados, es decir una medida de similitud entre **V** y **WH**. El resto de los términos en la función objetivo son elementos de regularización. $\|\mathbf{W}\|_1, \|\mathbf{H}\|_1$ son la L_1-norm [29] de **W** y **H** respectivamente y $\|\mathbf{W}\|_F^2, \|\mathbf{H}\|_F^2$ son la L_2-norm. Estos factores regularizadores son ponderados por dos parámetros: α, que afecta a todo los términos regularizadores; y ρ, que controla la prioriza la influencia de la norma L_1 o de la norma L_2. Cuando ρ es 1, solo los regularizadores basados en la norma L_1 son significativos en la función

[2] en machine learning a este concepto se le conoce como dispersión o *sparseness*.

de coste y se obtienen resultados más dispersos. Por el contrario, si ρ es 0, la función de coste solo se ve afectada por los regularizadores basados en la norma L_2, y por tanto se obtendrá consumos base más densos.

Por último, dentro de la explicación de la aplicación de métodos NMF a datos de consumos eléctricos se introduce como se genera la matriz $\mathbf{V}$ de entrada. Este aspecto, bien podría haberse introducido en la sección 2.2.2, pero se ha decido introducir en este apartado, ya que el lector en este punto ya tiene mayor conocimiento de cómo es el funcionamiento de este tipo de técnicas. Si consideramos los datos de consumos eléctricos una serie temporal $\{x(t)\}$, se necesita una reestructuración de los datos para formar la matriz $\mathbf{V}$ como la matriz de entrada al algoritmo NMF. Se propone reestructurar la secuencia, dividiéndola en N ventanas contiguas de longitud M:

$$\{\underbrace{x(0),x(1),\ldots,x(M-1)}_{\mathbf{v}_1},\ldots,\underbrace{x((N-1)M),x((N-1)M+1),\ldots,x(NM-1)}_{\mathbf{v}_N}\}$$

Las ventanas forman la matriz $\mathbf{V}$ de observaciones como sigue:

$$\mathbf{V}=\begin{pmatrix} x(0) & x(M) & \ldots & x((N-1)M) \\ x(1) & x(M+1) & \ldots & x((N-1)M+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(M-1) & x(2M-1) & \ldots & x(NM-1) \end{pmatrix}$$

Los consumos base obtenidos por NMF están estrechamente relacionados con la longitud de las ventanas así que, se deben elegir de manera que se obtengan componentes más interpretables. Las series temporales de consumos eléctricos en grandes edificios típicamente se estructuran cíclicamente por periodos de días, semanas, meses y años. Si el tamaño de la ventana es escogido acorde a estos periodos de tiempo, se obtendrán resultados más interpretables, ya que estos mantendrán la estructuración cíclica según a estos periodos, permitiendo una mejor asociación entre los patrones extraídos y eventos relevantes en la red. En otras palabras, se obtendrán diferentes patrones en las componentes resultantes si las columnas de $\mathbf{V}$ son elegidas como días, semanas o meses. Por consiguiente, NMF devolverá componentes con patrones a escala diaria, semanal o mensual.

Teniendo en cuenta el razonamiento anterior, se ha elegido $M = 1440$ de tal forma que una columna de $\mathbf{V}$ corresponde a los minutos de un día; de esta manera la matriz $\mathbf{V}$ estará compuesta por $N = 365$ observaciones [3] diarias de la demanda total del hospital.

[3] A las 365 observaciones hay que restarles los días en los que existen errores de medida.

Para realizar esta operación dentro de nuestro código, una vez más, recurriremos a Pandas y su función `pivot_table`. Esta potente funcionalidad de pandas, nos permite a partir de un DataFrame con múltiples atributos, generar otro, el cual tiene por índice y columnas todos los valores posibles de dos atributos del DataFrame inicial y como valores almacenados, un tercer atributo de la entrada. Imaginemos que tenemos un DataFrame, el cual tiene como atributos A, B y C, y generamos un *pivot table*, el cual tiene: por filas los posibles valores de A; por columnas los posibles valores de B y por elementos los valores de C. Extraer del nuevo Dataframe la fila con índice igual a i, es la misma operación que extraer las filas del Dataframe inicial cuyo atributo A sea igual a i. Lo mismo ocurre si eligiésemos una columna j, estaríamos filtrando el Dataframe por el atributo B. En caso de que existan varios valores de C para valores concretos de A y B, `pivot_table` genera un valor agregado de los mismos como nuevo elemento del DataFrame resultante. la función de agregación por defecto es la media, pero Pandas permite introducir nuestras propias funciones de agregación.

Volviendo a nuestro propósito de reestructurar la matriz $\mathbf{V}$, realizaremos una operación `pivot_table` acorde dos atributos: el minuto del día y el día del año y como valores a almacenar, los consumos eléctricos. De esta forma, estaremos generando una matriz de observaciones, que por filas tiene los consumos del día en minutos y por columnas los consumos diarios a lo largo de un año.

```
KwT_p = pd.pivot_table(data = KwT, index = 'Time', columns = 'DayOfYear',values
    = 'Valor' )
```

2.3.3 Implementación en python: Scikit-Learn

De vuelta a nuestra aplicación, la implementación de NMF, siguiendo la linea marcada por el resto apartados del trabajo, se realizará mediante Python. Uno de los atractivos de Python, es que existen módulos que abarcan casi la totalidad de las áreas de la ciencia. Como no podía ser de otra manera, existe un módulo específico para *Machine learning* llamado *Scikit-Learn*. "Scikit learn proporciona herramientas simples y eficientes para minería de datos y análisis de datos"[21]. Dentro de este módulo se encuentra una función para la aplicación de NMF, con las características expuestas en la sección anterior. La función NMF, devuelve un módelo de NMF acorde a una serie de parámetros como el número de componentes a estimar L, la inicialización, el número máximo de iteracciones y los parámetros propios de la regularización `alpha`(α) y `l1_ratio` (ρ). Un ejemplo de instancia de un modelo NMF se hace en el siguiente fragmento de código:

```
from sklearn.decomposition import NMF

n_comp = 4
n_iter = 500
l_ratio = 1
alpha = 1
```

```
model = NMF( n_components=nComp,random_state=None, init= 'nndsvd',max_iter=
    n_iter, solver = 'cd', l1_ratio=l_ratio, alpha = alpha)
```

Una vez creado el `model`, mediante su propio método `fit_transform`, *entrena* y *transforma* la matriz de observaciones **V**, devolviendo la matriz **W**:

```
W = model.fit_transform(Kw_v)
```

en nuestro caso, para próximos propósitos, también necesitamos la matriz de coeficientes **H**, que se extraerá mediante el método del modelo `componets_`:

```
H = model.components_
```

Estas dos matrices aquí obtenidas serán los datos de partida de herramientas de visualización que permitan al usuario, mediante su percepción visual, una comprensión más profunda de los resultados obtenidos. Todos los códigos de análisis NMF, realizados con scikit-learn, así como los realizados con Pandas para el pretratamiento serán recogidos por el archivo Python *NMFKwTot.py*.

2.4 Visualización

El valor de cualquier trabajo es proporcional, no solo a los resultados obtenidos, sino también la forma de presentarlos. Los resultados del trabajo hasta el momento, no son más que secuencias de números almacenados en variables, esta información se puede *codificar* de muchas maneras, por ejemplo, imprimiendo por pantalla esa secuencia de información en forma de números en el sistema decimal. Esta claro que este tipo de codificación no es la más adecuada para la percepción humana, ya que la comprensión de los resultados supondría un enorme esfuerzo mental. Es por tanto esencial para aumentar el valor del trabajo, encontrar una codificación de los resultados de forma que el proceso de comprensión de los mismos sea lo más ágil posible. En el presente trabajo esta codificación aprovechará uno de las mecanismos sensoriales más potentes del ser humano, la *percepción visual* [51]. En el resto de la sección será expuesta una codificación acorde a nuestro sistema visual, así como una serie de técnicas de visualización 2D que aprovechen esta codificación.

Una mera codificación visual de los datos es, una representación estática lo cual puede ser, según que aplicaciones, insuficiente para la percepción de los resultados, ya que el usuario quiere interactuar con la visualización, seleccionando, focalizando o filtrado lo que quiere ver. Estos elementos de *interacción*, potencian el proceso de comprensión y por tanto se añadirán en nuestra representación de los resultados.

2.4.1 Codificación visual

El paso de una codificación numérica a una codificación visual es delicado y se debe tener en cuenta la naturaleza de los datos a representar y que información queremos resaltar. Los resultados obtenidos son esencialmente la matrices de consumos base **W** y la matriz de coeficientes **H**. El perfil de consumo de un día en concreto estará compuesto por una combinación lineal de las columnas de **W**, cuyos coeficientes vendrán impuestos por las columnas de **H**. Si se analizan estos resultados de acuerdo con las posibilidades de visualización, se destaca por un lado, que portan un carácter temporal, ya que una columnas de **W** abarca un periodo de un día y que una fila de **H** abarca un año y, por otro lado, la información a presentar en cada día está formada por cuatro elementos distintos entre sí, es decir, tenemos datos categóricos. Esta reflexión se tendrá en consideración para las próximas decisiones en materia de visualización.

En el contexto de la codificación visual existen dos términos fundamentales, que condicionan las representaciones posteriores [44]:

- ***Marcas***: representan una muestra de los datos mediante una primitiva geométrica. En la figura 2.6, se muestran algunos ejemplos de marcas.

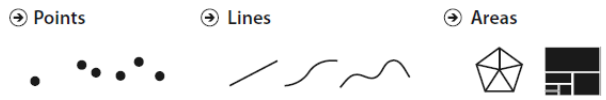

FIGURA 2.6. Ejemplos de marcas de codificación visual. (Fuente: [44])

- ***Canales***: son una característica de las marcas, la cual, cambia nuestra percepción de la misma, de acuerdo con la información que porta el conjunto de datos que representa. En la figura 2.7 se muestran diferentes ejemplos de canales.

De esta forma, un ejemplo de marca sería representar una medida mediante un círculo y como canal podría ser el radio, así, cuando la magnitud medida aumenta, el círculo aumentará de tamaño también. Según [44], el impacto de los diferentes canales y marcas en nuestra percepción no es el mismo, sino que, dependiendo de la información a transmitir, existen parejas de canales-marcas más efectivos que otros.

Si nos centramos únicamente en los canales, y teniendo en cuenta la naturaleza de los resultados NMF, uno de los canales más efectivos para representar datos temporales, es la *localización*. Tenemos la idea de que el tiempo "discurre", como que existe una "linea temporal" o "timeline", que en las representaciones temporales, transcurre en uno de los ejes, y generalmente

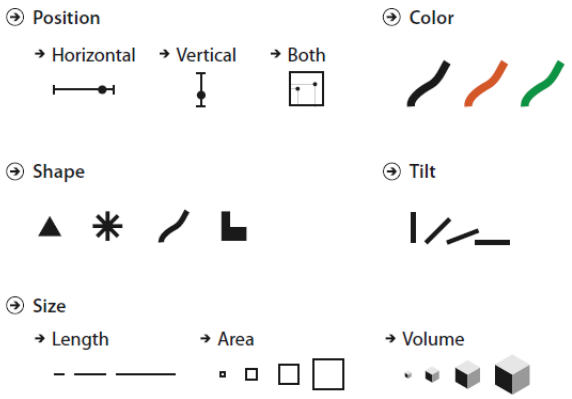

FIGURA 2.7. Ejemplos de canales de codificación visual. (Fuente: [44])

de izquierda a derecha o de arriba abajo. Otro ejemplo claro de representaciones temporales que usen el canal de la posición son los calendarios. Los calendarios son instrumentos visuales que habitualmente usamos y que la mayoría reconocemos, y no es más que una distribución espacial de las fechas de un año. Es, por tanto, conveniente en este trabajo el uso de la localización para la visualización de los resultados.

Otro de los canales más habituales es el uso de colores. Dentro del canal *color*, se pueden especificar los canales: de luminosidad; saturación de color (figura 2.8), que son habituales en la representación de *magnitudes secuenciales*; y la tonalidad que es uno de los canales más efectivos para la representación de datos categóricos. Un ejemplo de datos categóricos son las distintas componentes extraídas del NMF; por ello, para representar e identificar las distintas componentes base extraídas, se les asignará una tonalidad distinta a cada una de ellas. Todas las escalas de colores son obtenidas de la página [6].

FIGURA 2.8. Tipos de canales basados en el color. (Fuente: [44])

El uso de las marcas y canales, es fundamental en las diferentes *técnicas de visualización* aplicadas para generar una representación apropiada, que agilice el proceso de comprensión de la información.

2.4.2 Técnicas de visualización 2D

Los resultados portan información *cuantitativa secuencial*, es decir, los consumos base obtenidos y sus coeficientes son magnitudes que se pueden cuantizar en el sistema decimal y sus valores son siempre positivos. Dentro de las representaciones cuantitativas las técnicas más eficientes son las visualizaciones 2D [49], de entre las cuales, de acuerdo con el tipo de datos y haciendo uso de los canales anteriormente mencionados, se han elegido las siguientes técnicas para nuestra representación:

- ***Calendario***: El calendario es una disposición espacial, que utiliza una organización por todos conocida para representar datos anuales. Sabiendo que las filas de la matriz **H** contienen los coeficientes de todo un año para una de las componentes extraídas, usar un calendario parece una buena opción para representar la magnitud de estos coeficientes a lo largo del año. De esta forma, se muestra al usuario la influencia de la componente a lo largo de un año mediante en una representación que ya tiene interiorizada.

  Entrando en aspectos de diseño del calendario, a los días se les asocian como marcas rectángulos y como canal proporcional a la magnitud de los coeficientes se propone el uso de la saturación del color. La posición se organiza horizontalmente según la semana del año y verticalmente según el día de la semana. No existe separación entre los meses, sino que se resalta con un contorno mas grueso cada mes, esta forma de representación continua de todos los días permite apreciar con mayor fuerza los patrones semanales o por los días del año, ya que esta continuidad aumenta el estímulo percibido según la Ley de Weber [30]. Un ejemplo de este tipo de representaciones es la figura 2.9

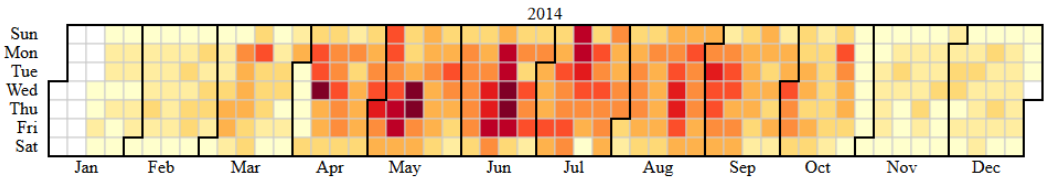

FIGURA 2.9. Ejemplo de la representación calendario.

- ***Stacked area***: esta representación, que se basa en gráficos de línea o *linechart*, como el de la figura 2.10, no es más que apilar verticalmente varios de estos gráficos, identificando mediante una escala de color categórica, las áreas formadas entre dos tendencias contiguas. Los gráficos *stacked graph* [35] son muy utilizados para representar series temporales que son resultado de una agregación de varios elementos. Si identificamos estos elementos como los consumos base multiplicados por los coeficientes propios de un día, la agregación de los mismos formarán una serie temporal que abarca un día. Entonces podemos adoptar

la técnica Stacked área como una técnica adecuada para representar los consumos desagregados como partes de un consumo total. Este tipo de representaciones transmiten a la perfección la influencia de cada una componentes en el consumo total a lo largo de un día, como se puede observar en la figura 2.11. Es importante destacar, que si representásemos cada componente por separado mediante un linechart obtendríamos perfiles distintos, esto se debe a que solo la primera área en apilar, tiene como línea base el eje horizontal; el resto, tienen como línea base el perfil del área inmediatamente inferior. Esto, provoca que este tipo de representaciones pierdan eficacia en comparar las partes del total entre sí, forzando a incorporar en nuestra aplicación mecanismos para paliar estas desventajas, como se expondrá más adelante.

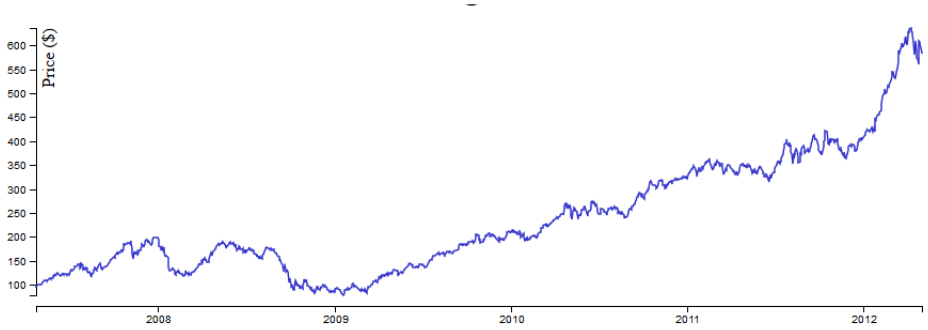

FIGURA 2.10. Ejemplo de la representación line chart.

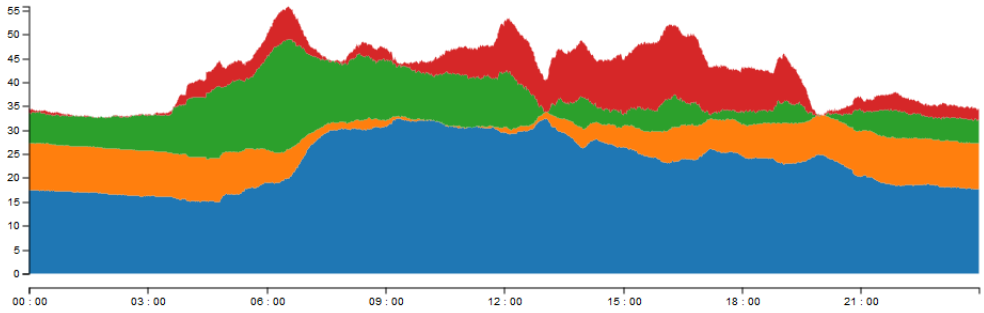

FIGURA 2.11. Ejemplo de la representación stacked area.

Estas dos representaciones, hasta el momento, son estáticas; solo podemos representar los coeficientes de una componente en el calendario, y solo el consumo desagregado de un único día. Que el usuario pueda elegir qué día quiere explorar o qué coeficiente representar en el calendario es una de las tareas que añadirían dinamismo y valor a nuestra visualización. En el

siguiente apartado, se desarrollarán los criterios de interacción adoptados para la consecución del dinamismo propuesto.

2.4.3 Métodos de interacción

La mayoría de las aplicaciones visuales modernas están basadas en dos ejes fundamentales: la *representación* y la *interacción*. La representación, estará formada por las técnicas de visualización mencionadas en la sección anterior. Por su parte, la interacción engloba el dialogo entre el usuario y la propia aplicación, de tal forma que tenga la posibilidad de explorar los resultados según sus inquietudes. Dentro de este concepto de interacción se desprenden siete categorías [53], de las cuales, nos centraremos en la selección. Este tipo de interacción es aquella en la que el usuario indica qué parte de la representación considera interesante para sus propósitos, mostrándose una información más detallada sobre los elementos indicados. En la aplicación se habilitan dos tipos de selección:

- ***Selección del día***: selección el día del calendario que el usuario desea obtener los perfiles desagregados mediante el gráfico Stacked. Esta selección vendrá desencadenada por un evento "click" o "mouseover" sobre la marca del calendario que representa el día seleccionado. Tras este evento se resaltará dicha marca sobre las demás de algún modo, para que el usuario tenga constancia de lo que ha seleccionado y qué día es el correspondiente al stacked representado.

- ***Selección de la componente***: si dentro del gráfico Stacked seleccionásemos un área, el usuario está mostrando predisposición de analizar esa componente y por tanto quiere modificar la visualización de acuerdo a ella. Esta modificación se materializa en mostrar el calendario asociado a la componente seleccionada y en modificar el gráfico Stacked, de tal forma que dicha componente sea la referida al eje horizontal.

Hasta ahora, se ha explicado los conceptos de la visualización de los resultados, pero no se ha tratado cómo se van ha convertir estos conceptos en una aplicación real y funcional, por ello, en la siguiente sección 2.6 se expondrá una implementación basada en entorno web basada en el patrón MVC.

2.5 Model View Controller

En el capítulo 1 se propuso la intención, de basar nuestra aplicación en el modelo de MVC (figura 1.4). Este patrón para el diseño de interfaces gráficas de usuario, se compone de tres pilares:

- ***Model***: el modelo gestiona la información introducida por el usuario, comprobando que se encuentra dentro de las especificaciones del sistema y enviándosela a "View".
- ***View***: tras recibir la información de "Model" se encarga de generar, generá una visualización adecuada a ella y que facilite la interacción con el usuario.
- ***Controller***: es el encargado de recibir los eventos por parte del usuario y el intermediario en la trasiego de datos entre "Model" y el usuario que ha percibido la visualización.

La separación de estos tres bloques pretende separar los elementos meramente representativos de la interacción del usuario. Basándose en este patrón, se modifica el esquema general de la figura 1.4, por uno más específico como el de la figura 2.12, que se amolde a las intenciones finales del presente trabajo. Básicamente se ha añadido el bloque "Compute" como un nuevo elemento en el ciclo, encargado de realizar los cómputos del análisis inteligente y de generar una nueva información a "View", este nuevo módulo podría estar incluido dentro de "View", pero se ha decidido separarlo para una mejor organización conceptual.

Nuestro "Controller" recibirá eventos por parte del usuario que, tras ser verificados por "Model", modifiquen parámetros del análisis generando nuevos resultados NMF, que a su vez son la base de una nueva realimentación visual al usuario, por parte de "View".

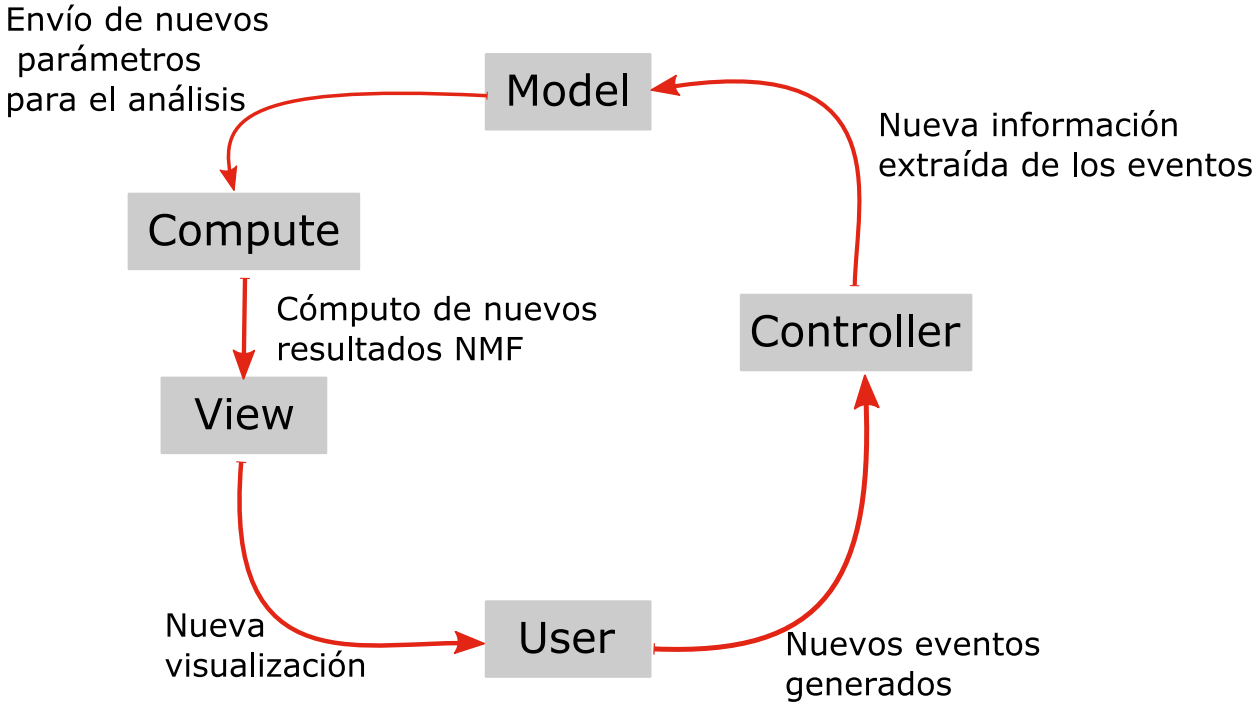

FIGURA 2.12. Modelo MVC adaptado.

Hasta ahora se han desarrollado los métodos conceptuales de "View" y "Compute" en las secciones 2.3 y 2.4 respectivamente, pero nada se ha mencionado de cómo agrupar estos dos pilares mediante Controller y Model, ni de cómo llevar a la práctica estos conceptos con herramientas software. Partiendo de que la aplicación se ha decidido llevar a cabo en un entorno web, existen *framework* [4] específicos para este desarrollo, inspirados en el modelo de la figura 2.12, y además basados en Python. Entre ellos, uno de los más sencillos e inmediatos en el desarrollo es *Flask*, cuyo uso para el desarrollo de nuestra aplicación viene explicado en la siguiente sección.

2.6 Entorno web: Flask, HTML, CSS y Javascript

Se ha decidido hacer el desarrollo de la aplicación en un entorno web debido, entre otras razones, a la enorme potencia y portabilidad que presenta *HTML5* [27] y el poder que nos ofrece *Javascript* [28] para realizar visualizaciones dinámicas dentro de HTML5 mediante potentes librerías como D3 (Data Driven Document) [7].

Una vez tomada las decisión de basarse en HTML y Javascipt para la parte visual de la aplicación y seguir un patrón de diseño MVC, se debe escoger un *framework* para el desarrollo de las partes restantes del modelo y que ayude a combinar los elementos del mismo entre sí. El pretratamiento y el análisis de los datos, forma parte de la etapa "Compute" del patrón y se ha llevado a cabo mediante Python; por ello, parece adecuado que el framework MVC elegido esté basado en Python. De todos los frameworks que reunían estas características, se ha elegido *Flask* [11], por: su sencillez en el desarrollo; su gran potencia y flexibilidad en la creación de platillas HTML; y la gestión de los eventos generados por el usuario. En lo que queda de sección, se explica el uso de Flask a partir de: su organización de archivos; cómo recoge, comprueba y reacciona ante eventos por parte del usuario; y por último la creación de plantillas HTML y el uso de Javascript dentro de ellas.

2.6.1 Organización de archivos

En nuestro directorio raíz de trabajo en flask es aquel donde se encuentre el archivo Python donde se importe el módulo Flask, el cual hará de "Controller" dentro del patrón MVC. En este directorio, como se muestra en la figura 2.13, típicamente se encuentran, además del mencionado archivo de python: las carpetas *static* y *templates*. Como la aplicación se basa en el modelo MVC los archivos de "Compute", "Model" desarrollados en Python han de ser incluidos en el directorio raíz:

- ***controller.py***: recoge las instrucciones propias del controlador en el modelo MVC y es el

[4] En desarrollo de software es un marco de trabajo, soportado por un conjunto de programas y bibliotecas que está orientado a para agilizar el desarrollo de aplicaciones con una funcionalidad específica.

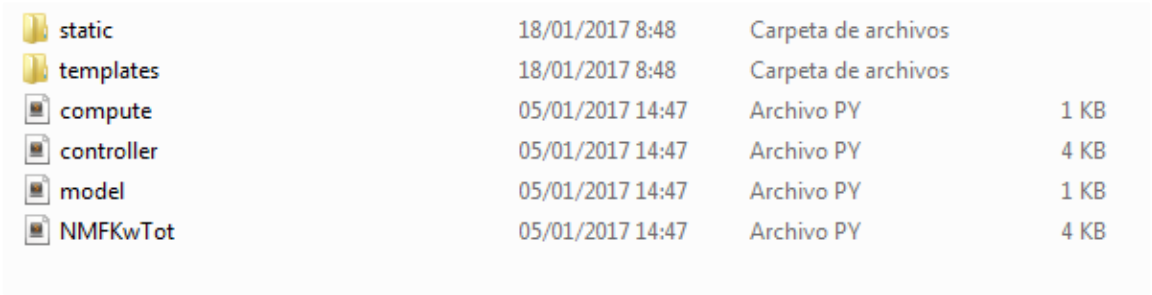

FIGURA 2.13. Detalle del directorio raíz de Flask.

eje central de toda la aplicación. Se encarga, como se explicará con más detalle, de recoger los eventos, y de hacer las llamadas pertinentes a "Model" y "Compute".

- ***compute.py***: es el archivo que contiene las funciones necesarias para llevar a cabo las tareas de análisis y pretratamiento de los datos. En etapas iniciales del desarrollo del trabajo, previas a la unión de todos los códigos en una misma aplicación, se desarrollaron funciones para el análisis en forma de módulo Python recogidas en el archivo *NMFKwTot.py*, el cual será importado en el archivo *compute.py* para el acceso a dichas funciones.

- ***model.py***: es el encargado de comprobar la información que portan los eventos, importa módulos específicos de Python para dicha tarea. En especial, se hará uso del módulo *wtforms*, el cual es especificó para el parseo de peticiones POST y GET de HTML [31]. Dentro de un *framework* web, como es el caso de Flask, los eventos más comunes son las peticiones GET POST. En el caso concreto de nuestra aplicación, este tipo de eventos se usarán para enviar parámetros del análisis como el número de componentes a calcular.

- ***static***: en esta carpeta se almacenan los archivos que necesitan las plantillas HTML, es decir: archivos de datos, hojas de estilo CSS y todos los archivos Javascript que manejan las visualizaciones interactivas. Estos últimos, debido a su complejidad se explicarán en la subsección 2.6.4 explicando la metodología adoptada para la programación de la parte visual de la aplicación.

- ***templates***: en esta carpeta se alojan los archivos HTML, que conforman las plantillas flask. El término plantilla es usado para referirse a documentos HTML, que el motor flask es capaz de renderizar y servir en una un dirección IP y en un puerto concreto. Además, las plantillas flask están basadas en *Jinja2* [14] un motor de plantillas que permite el paso de ciertos parámetros a dichas plantillas, permitiendo modularidad en el código, incluso herencia entre las plantillas como se explicará en la subsección 2.6.3.

2.6.2 Controller

El archivo *controller.py* es el eje central de la aplicación; en él, se encuentra la llamada al objeto `Flask`, encargado de servir la aplicación. A través de este objeto, podemos instar *listeners*, los cuales no son más que *decorators* [8] de python, que a efectos prácticos asocian una URL concreta con una función de python. En el siguiente fragmento de código se muestra la instancia del objeto *Flask* y la creción del listener de nuestra aplicación *listeners*:

```
from flask import Flask, render_template, url_for, request
import os
import compute
from model import InputForm

app= Flask(__name__)

# Posibles direcciones

# Direccion raiz muestra un calendario dinamico

@app.route('/',methods=['GET', 'POST'])
def dinamico():

    form = InputForm(request.form)

    # En la primera carga rederiza unos datos pre calculados
    if (request.method == 'POST' and form.validate()):
        datacomputo_h, datacomputo_w = compute.ComputeNmf(nComp=form.N.data,
                                pathData = "./static/data/KwTEneroDiciembre.csv
                                    ",
                                l_ratio = form.l_ratio.data,
                                n_iter = form.n_iter.data,
                                alpha = form.alpha.data)

    else:
        datacomputo_h, datacomputo_w = compute.parseo()


    return render_template('dinamico.html', style = url_for('static', filename
        = 'style.css')
                                ,D3Calendar = url_for('static',filename = '
                                    calendar.js')
                                ,stacked = url_for('static', filename = '
                                    stackedArea.js')
                                ,Caller = url_for('static', filename = 'caller.
                                    js' )
                                , Jdata_h = datacomputo_h
                                , Jdata_w = datacomputo_w
```

```
, form = form)
```

cuando al motor Flask le llegue una petición con la ruta "/"se ejecutará la función `dinamico`, en ella se hace uso de elementos de *model*, *compute* y el renderizado de las plantillas. Lo primero que nos encontramos en la función `dinamico`, es una instancia a InputForm, que no es más que una clase propia que valida la información procedente de los métodos POST. Dependiendo de si se produce un evento POST y de su comprobación, se generan unos datos u otros que serán los parámetros de las plantillas a renderizar. En el caso de que sea la primera vez que se carga la aplicación, es decir no se ha producido ningún evento POST, se hace la llamada a la función `parseo()` de *compute.py* que devuelve resultados NMF precalculados con los que poder renderizar una página inicial. Por el contrario, si se produce un evento POST, es decir, el usuario ha cambiado algún parámetro en la interfaz, se produce una llamada a la función `Computenmf()` que calcula unos nuevos resultados NMF acorde a los nuevos parámetros enviados y renderiza una nueva visualización acorde a ellos.

A la hora de renderizar las plantillas Flask, se cuenta con la función `render_template`, la cual tiene como parámetros los datos calculados y los archivos Javascript y CSS. Estos últimos, son pasados como una *url* del path donde se encuentran y para crearla se hace uso de la función `url_for()`.

2.6.3 Plantilla HTML

Dentro de la carpeta de *templates* se encuentran los archivos HTML correspondientes a las plantillas renderizadas por *controller.py*, estas, no son más que documentos html en cuyo interior se encuentran porciones de código con sintaxis propia de *Jinja2*, que permiten definir la herencia y los parámetros de las plantillas. Que Flask permita el paso de parámetros a las plantillas añade modularidad a los diseños, además de dinamismo ya que pueden cambiar su apariencia con solo cambiar algún parámetro.

En cuanto a la herencia, permite encapsular los códigos permitiendo generar programas facilmente reutilizables en otras aplicaciones o dentro la misma. Por ejemplo, imaginemos una página web con varias pestañas por las que podamos navegar entrando en distintas subpáginas; normalmente en este tipo de aplicaciones la cabecera de la página principal permanece invariante a lo largo de las subpaginas, por tanto, si desarrollasemos estas aplicaciones sin plantillas, el código HTML asociado a la cabecera se repetiría en todos los HTML de cada subpágina, mientras que con la herencia solamente necesitaríamos una plantilla base que recogiera esa cabecera y el resto de plantillas que heredarían de ella.

En nuestra aplicación, solo existe una plantilla a renderizar, pero aún así, se ha decidido generar una plantilla base para recoger la cabecera, pensando en posibles ampliaciones de la

aplicación. Teniendo en cuenta esto, dentro de *templates* existen dos plantillas: una plantilla base *layout.html* y una plantilla principal denominada *index.html*. Para la comprensión de estas plantillas, es posible que el lector necesite consultar [14] para informarse sobre la sintaxis para la introducción de parámetros y herencias.

2.6.4 Archivos Javascript

El uso de Javascript para la representación visual de nuestra aplicación, es debido principalmente a dos motivos principales: la potencia de cálculo que desarrolla Javascript al ejecutarse en el navegador y debido a la existencia de la librería javascript D3 (Data Driven Document). Esta librería, ofrece herramientas para el manejo de elementos elementos gráficos "svg", control de eventos desde el propio navegador, y lo más importante, nos permite vincular fácilmente estructuras de datos tipo *array* con elementos gráficos [12]. El hecho de tener cada dato, susceptible de representar, asociado a un elemento gráfico permite generar y eliminar marcas visuales, además de modificar alguno de sus canales acorde al dato asociado. Si tenemos en cuenta los conceptos planteados en el capítulo de visualización, podemos afirmar que D3 es una potente herramienta de codificación visual, con la que perfectamente podemos llevar a cabo la visualización iteractiva propuesta en ese mismo capítulo. La visualización propuesta contaba con un *calendario* y un *stacked area*, además de los elementos de interacción. Todo ello, se desarrollará bajo Javascript y D3 como se resume en lo que queda de sección.

Esencialmente la parte Javascript se compone de tres archivos: *caller.js*, *calendar.js* y *stacked.js*. Los dos últimos reúnen los códigos de las dos visualizaciones propuestas, y *caller.js* hace uso de ellas, gestionando dónde se dibujan y cómo se gestionan los eventos producidos entre ellas. El motivo de que estén en archivos separados las dos visualizaciones, es que para implementarlas se siguió una metodología aconsejada por *Mike Bostock*, creador de la librería, que garantiza la modularidad y permite generar visualizaciones interactivas reutilizables [25]. Estas visualizaciones están encapsuladas en *"closures"* de Javacript, que tienen como parámetros un elemento *div* de HTML y los datos a representar. En el siguiente fragmento de código se presenta la llamada de las dos representaciones:

```
var calendario = calendar();
var stacked = stackedArea().dinamic(true);

d3.select("#Calendar").datum(datavis_h[d_keys[0]]).call(calendario);
d3.select("#stacked").datum(datavis_w).call(stacked);
```

los argumentos `datavis` hacen referencia a los datos de las matrices **W** y **H** y los identificadores `#Calendar` y `#stacked` corresponden a los contenedores *div* que alojarán las representaciones. Las funciones `calendario` y `stacked`, contienen métodos para actualizar los datos asociados a sus elementos gráficos, capaces de actualizar toda la visualización acorde a ellos. Las llamadas a

estos métodos se ilustran en el siguiente código:

```
stacked.updateData(stackedData);
calendario.updatedata(datacomponente);
```

donde `stackedData` y `datacomponente` corresponden a los datos encargados de actualizar los elementos gráficos de las representaciones.

El código restante se encarga de realizar el formateo de los datos, acorde a las especificaciones de las funciones reutilizables, y de realizar la gestión de los eventos, incluyendo la multiplicación de los coeficientes por los consumos base para generar los datos del stacked area cada vez que se selecciona un día.

CAPÍTULO 3

RESULTADOS

En este capítulo se pretende, por un lado, mostrar la materialización de todos los conceptos explicados en el capítulo anterior en una aplicación visual y, por otro lado, presentar una serie de casos de estudio. En estos últimos se explican los patrones de consumo descubiertos, lo cuales pueden ser asociados a eventos características dentro de la red eléctrica del hospital.

3.1 Aplicación final

Todos los conceptos explicados anteriormente se han integrado en una aplicación visual e interactiva gracias a las herramientas software indicadas. En esta sección se cree conveniente explicar detalladamente de qué elementos consta esta aplicación y cómo se deben usar para la exploración de los resultados NMF. Esta sección es importante para el lector, ya que constituye guía imprescindible para comprender los casos de uso.

grupo de supervisión y diagnóstico de procesos industriales

GSDPI

NMF aplicado a datos eléctricos.

Aplicación interactiva basada en D3 y Flask para la exploración de los resultados de algoritmos de desagregación de energía basados en Non Negative Matrix Factorization. La representación tiene como datos de partida la potencia eléctrica en el punto de facturación del hospital de León. Esta información ha sido cedida por el grupo SUPPRESS de la Universidad de León.

En la primera carga de la página, la visualización mostrada se corresponde a unos resultados precalculados acorde a los parámetros indicados; no obstante, la aplicación consta de un apartado para la personalización de los resultados NMF, donde se puede especificar parámetros el algoritmo tales como: el número de componentes a calcular, el número de iteraciones (Epoch), el peso de la L1 Norm y alpha. Una vez seleccionados estos valores, el usuario debe pulsar *Compute* para generar los nuevos resultados.

Tras el cálculo de los resultados, es posible explorarlos a través de la visualización formada por un calendario y un gráfico *stacked*. Ambos elementos gráficos son interactivos y están coordinados entre sí, de forma que si usuario pasa el puntero por los días del calendario podrá observar el perfil de consumo desagregado al día correspondiente. En el caso de que se el usuario desee analizar con más detalle el consumo de un día en concreto podrá hacer *click* sobre él y la representación del perfil permanecerá fija, aunque el puntero cambie de día. Para liberar el perfil seleccionado, el usuario solo tendrá que hacer *click* de nuevo en el mismo día. En cuanto a la representación del perfil de consumo diario se podrá cambiar el área base haciendo *click* sobre el área deseada. A la vez que se produce el cambio de área base, el calendario también se actualizará para mostrar los coeficientes asociados a esa componente.

Parámetros NMF

Number of components	4
Epoch	50000
L1 ratio	1.0
Alpha	10

Compute

Visualización

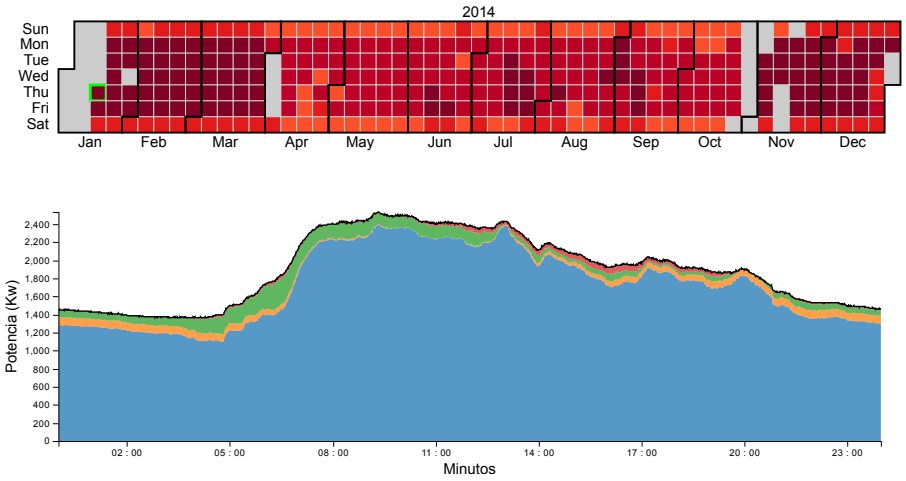

FIGURA 3.1. vista general de la aplicación.

3.1.1 Elementos de la aplicación

Una perspectiva global de la aplicación se muestra en la figura 3.1. A la hora de explicar la herramienta desarrollada al usuario se dividirá en tres partes, cada una de ellas con una funcionalidad distinta para el usuario:

- ***Introducción***: está compuesta por un texto introductorio,que contiene las instrucciones de uso básicas de la aplicación. En la figura 3.2, se muestra en detalle el texto introductorio.

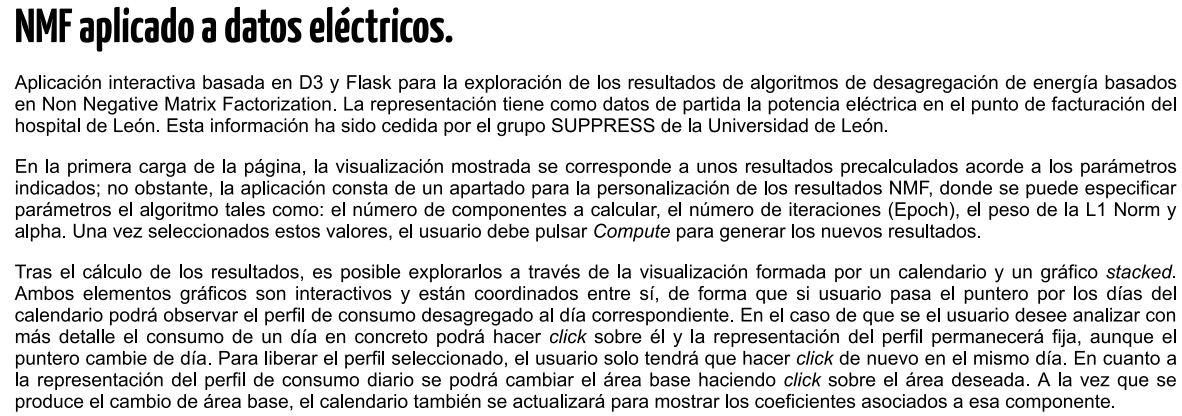

FIGURA 3.2. Detalle del texto introductorio.

- ***Parámetros***: en esta parte de la interfaz el usuario es capaz de introducir nuevos parámetros para un nuevo cómputo del algoritmo. Se compone, como se puede observar en la figura 3.3 por una serie de campos numéricos, donde el usuario puede indicar los siguientes parámetros del algoritmo NMF: el número de componentes a calcular, el número de iteraciones, el peso de la $L1\ norm$ y $alpha$. Dentro del modelo MVC, esta parte de la aplicación es donde el usuario puede generar los eventos que desencadenan un nuevo computo y el posterior cambio en la visualización.

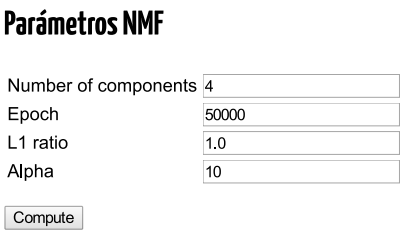

FIGURA 3.3. Detalle de la configuración algoritmo.

- ***Visualización***: se encuentra en la parte inferior de la página y está formada por las dos técnicas de visualización interactiva indicadas en el capítulo anterior: el calendario y el gráfico stacked area. Se decidió disponerlos uno debajo del otro en vez de uno al lado del otro, por dos motivos; uno, porque las dos representaciones ocupan bastante espacio horizontal y podría generar problemas en pantallas de ciertos dispositivos; y dos, si representamos los dos gráficos horizontalmente es posible que el usuario confundiera la magnitud de la potencia con los días de la semana. Una vista más detallada de las visualizaciones es presentada en la figura 3.4

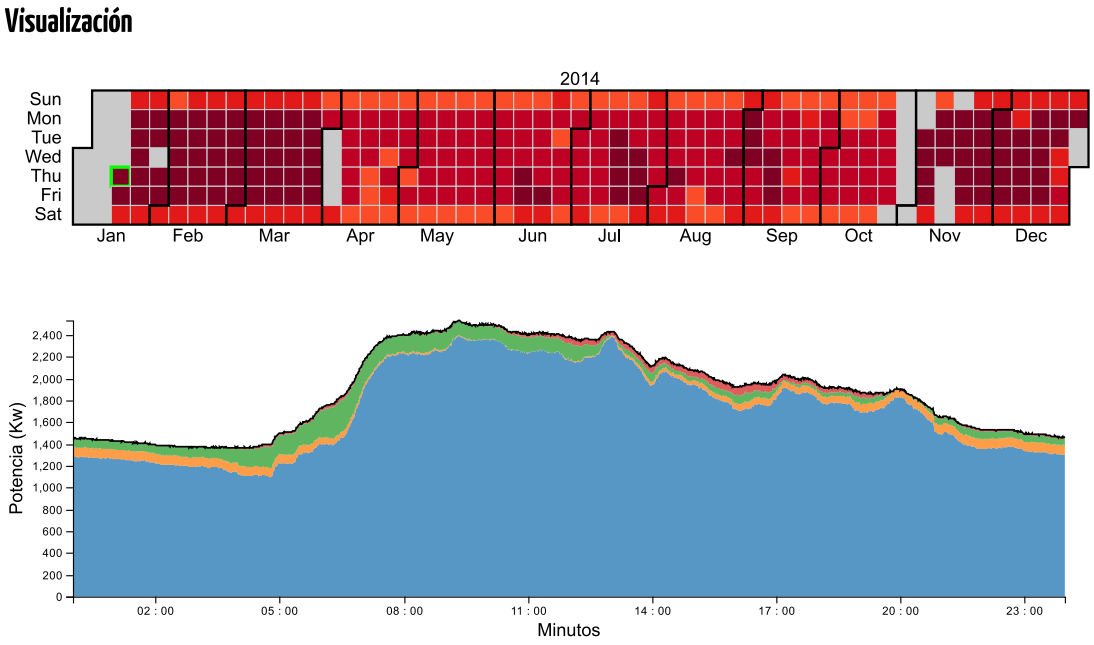

FIGURA 3.4. Detalle del texto de las visualizaciones.

3.1.2 Uso de la aplicación

En el texto introductorio de la aplicación se incluye un resumen de uso de la aplicación, pero aquí se explica detalladamente el uso y las funcionalidades que aporta esta aplicación.

En primer lugar, explicar que el desarrollo de la aplicación es completamente modular y que por tanto tras un nuevo cálculo NMF, aunque el número de componentes varie, las visualizaciones se adaptarán a esta nueva situación. Para generar un nuevo cómputo, el usuario solo tiene que introducir los nuevos parámetros (figura 3.3) y hacer "click" en el botón *compute* y Flask iniciará el cómputo NMF. La operación de cómputo puede tardar varios minutos, en los cuales el navegador indicará que está cargando la nueva página. Durante este periodo de tiempo el motor

Javascript sigue funcionando, de forma que las interacciones propias de las visualizaciones y por tanto el uso de la visualización no se ve afectado. En el instante en el que "compute" finaliza el cálculo, Flask envía la nueva página con los nuevos resultados.

En cuanto a la interacción, la aplicación está diseñada de tal forma que el usuario pueda generar eventos desde las dos representaciones y que tras un evento, sin importar su procedencia, las dos visualizaciones reaccionan ante él. Según este diseño tendremos dos *vistas coordinadas*, en las que podemos hacer una selección del día o la componente que se desea estudiar. Para seleccionar qué perfil de consumo desagregado diario se quiere representar en el stacked, existen dos posibilidades, pasar el puntero por encima de los días o hacer "click" sobre uno de ellos. La primera de ellas está pensada para poder deslizar el puntero a lo largo de los días y percibir el cambio entre ellos de una forma dinámica; mientras que la segunda es necesaria cuando se quiere observar en detalle el perfil. Cuando se hace "click" en un día, la aplicación deshabilita los eventos "mouseover", bloqueando el perfil mostrado. En esta situación el usuario puede cambiar el área base del gráfico stacked, mediante un "click" en el área deseada. Ante este evento también reacciona el calendario mostrando la influencia de esa componente a lo largo del año. En caso de que el usuario desee liberar el día seleccionado debe volver a hacer "click" sobre el mismo día o sobre otro cualquiera. Si lo hace sobre el mismo, la aplicación habilita de nuevo el evento "mouseover", mientras que si se hace "click" sobre otro se seleccionará ese día pero no se desbloqueará el evento "mouseover". Un ejemplo de uso siguiendo esta secuencia se muestra en la figura 3.5

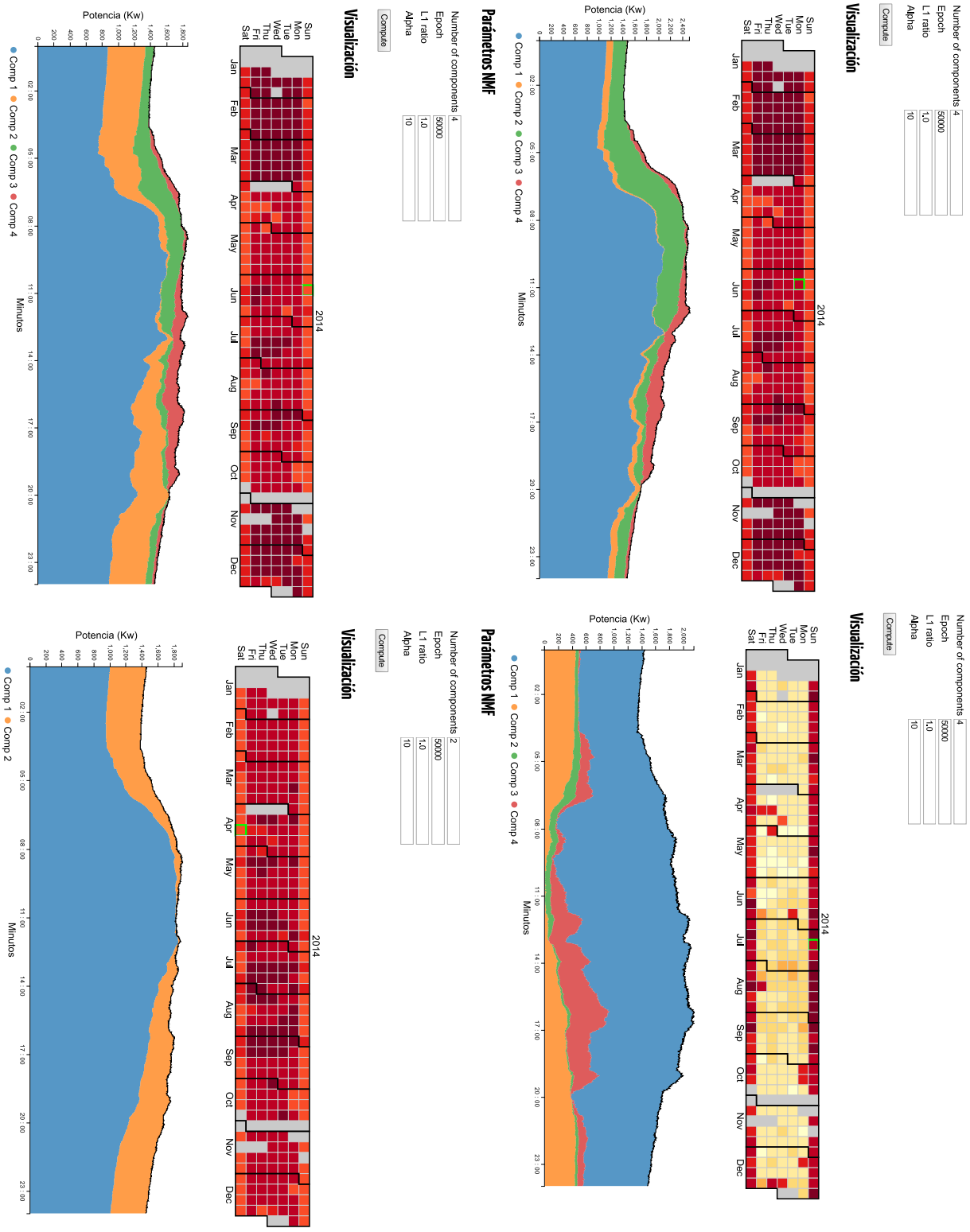

FIGURA 3.5. Imágenes de diferentes posibilidades de uso.

3.2 Casos de estudio

Los casos de estudio planteados consisten en identificar consumos base que podamos relacionar con comportamientos conocidos en la red, y probar de esta manera la efectividad del algoritmo NMF para la desagregación de la energía. Se mostrarán cinco patrones, los cuales pueden ser contrastados con eventos característicos de la red. A la hora de obtener estas componentes se indicará con qué parámetros del algoritmo se han conseguido, de entre los cuales se hace hincapié en el número de componenentes calculadas. Los casos estudiados se mencionan en la siguiente lista:

1. Consumo base asociado a días laborables.

2. Consumo base asociado a días festivos.

3. Consumo base asociado al arranque de la parte de frío en sistemas HVAC.

4. Consumo base asociado a grandes consumos a partir del mediodía.

5. Consumo base asociado a baja demanda de frío.

Todas estos consumos base han sido identificados gracias a la ayuda de profesionales tanto del Hospital de León como del grupo SUPPRESS, quienes poseen un conocimiento amplio del sistema analizado.

3.2.1 Consumo base asociado a días laborables

Para la obtención de este consumo base y de los tres siguientes se configurará el algoritmo NMF como se muestra en la figura 3.6. La primera de las cuatro componentes presenta un calendario y un perfil diario como el mostrado en la figura 3.7. Como se puede observar, tiene una fuerte influencia a lo largo de todo el año, pero, se percibe que los días no laborables el valor de los coeficientes disminuye, por ello, se ha asociado esta componente a los días laborables, ya que es en estos cuando su influencia es máxima.

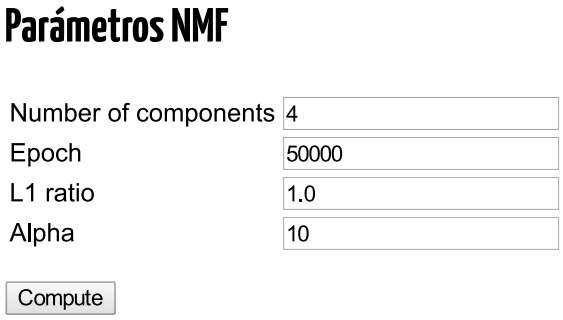

FIGURA 3.6. Configuración del algoritmo para los cuatro primeros casos de estudio.

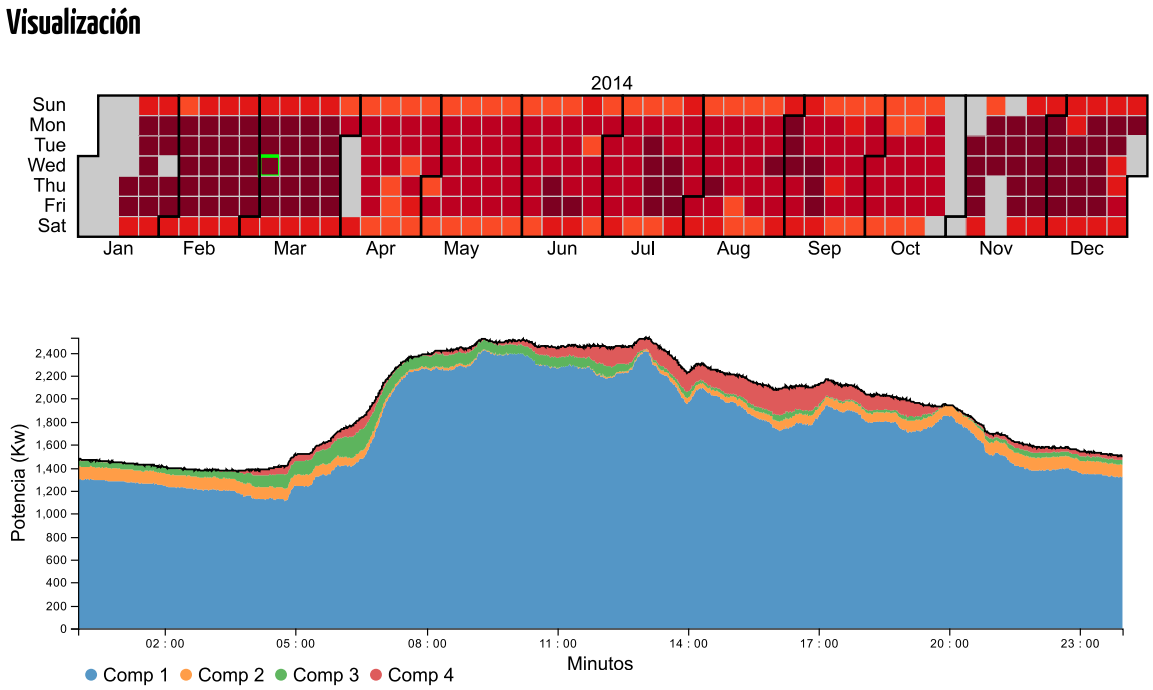

FIGURA 3.7. Visualizaciones asociadas a la componente laboral.

En la explicación de los fundamentos del algoritmo se afirmó que cuanto más dispersos fueran los resultados mejor sería su interpretación, pues bien, esta componente, de entre todas las estudiadas, es la única que es no es dispersa ni en los componentes, ni en el consumo base obtenido. La conclusión extraída tras este resultado es que es un consumo principal, en el cual

se le agregarán peculiaridades. Si nos centramos en su perfil por minutos, la silueta coincide con la que un humano representaría para resumir todos los perfiles de consumo analizados, es decir, es aquella que podríamos usar como lienzo para añadir otros consumos base dispersos para componer perfiles diarios concretos. En cuanto a su forma es una joraba con su mayor valor al medio día que va disminuyendo a lo largo de la tarde.

3.2.2 Consumo base asociado a días festivos

Manteniendo la configuración de la figura 3.6, el segundo consumo base obtenido y su correspondiente calendario se muestra en la figura 3.8. En esta componente se observa claramente que su influencia es superior en los fines de semana y en una serie días, los cuales coinciden con los días festivos en la ciudad de León. Se puede afirmar, por tanto, que el consumo base obtenido identifica días no laborables de la ciudad de León.

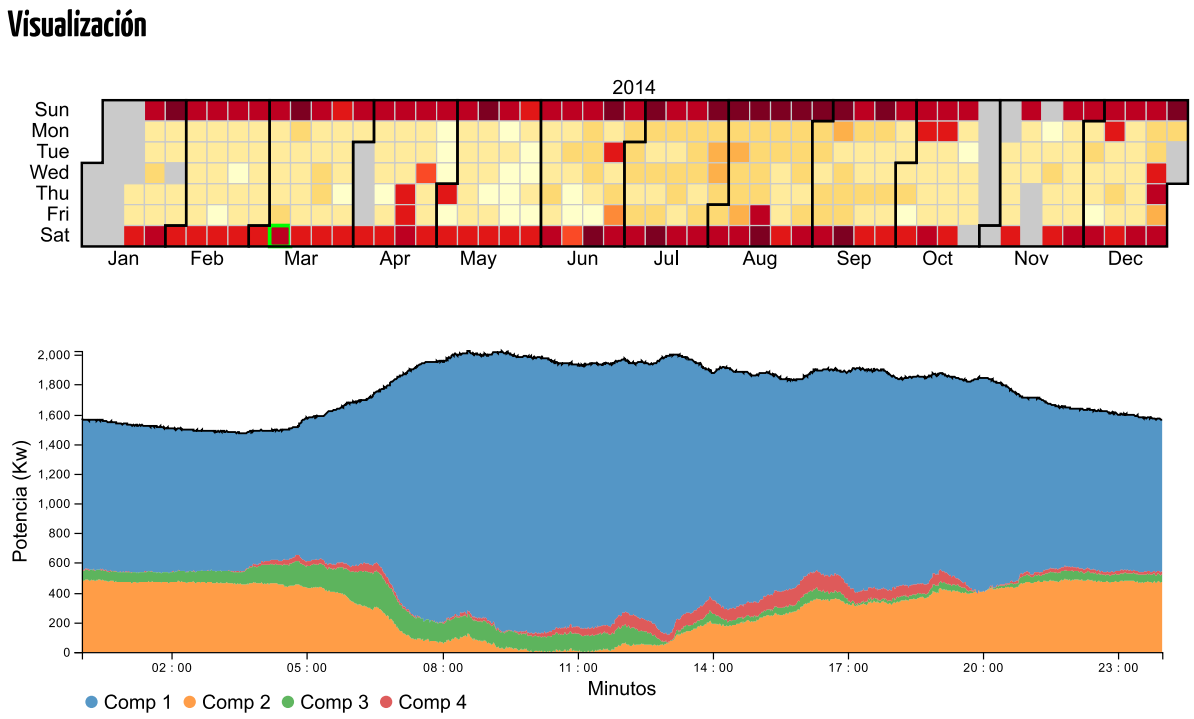

FIGURA 3.8. Visualizaciones asociadas a la componente festiva.

En las jornadas que resalta esta componente, el hospital se comporta de forma distinta ya que el número de consultas es reducido y solamente tiene actividad la zona de hospitalizaciones. Esto se materializa en la demanda diaria, en un suavizado de la joroba dando como resultado una forma mucho más plana. Si partimos del consumo base anterior, para formar un perfil más aplanado se debe agregar un consumo más pronunciado a primera hora de la mañana y última

hora de la tarde, como el presentado por esta segunda componente como el gráfico stacked de la figura 3.8.

3.2.3 Consumo base asociado al arranque de la parte de frío en sistemas HVAC

La tercera de las cuatro componentes extraídas según la configuración de la figura 3.6 genera la representación mostrada en la figura 3.9. La influencia a lo largo del año de esta componente revela un claro comportamiento estacional, siendo de mayor importancia en la primavera y el verano. Al contactar con el personal del grupo SUPPRESS, asociaron ese comportamiento al arranque de la parte de frio de los sistemas HVAC. El personal técnico del hospital, deshabilitan la parte de frio de estos sistemas durante el invierno, activándola a principios de abril y la deshabilitándola a finales de octubre, lo cual coincide con los resultados obtenidos. Además, estos sistemas arrancan a las 6:00 para que estén operativos a la apertura de las consultas a las 8:00, lo cual coincide con la perfil representado en el gráfico stacked que presenta un pico de consumo a esa hora. Con toda esta información podemos identificar esta componente con el consumo de estos sistemas.

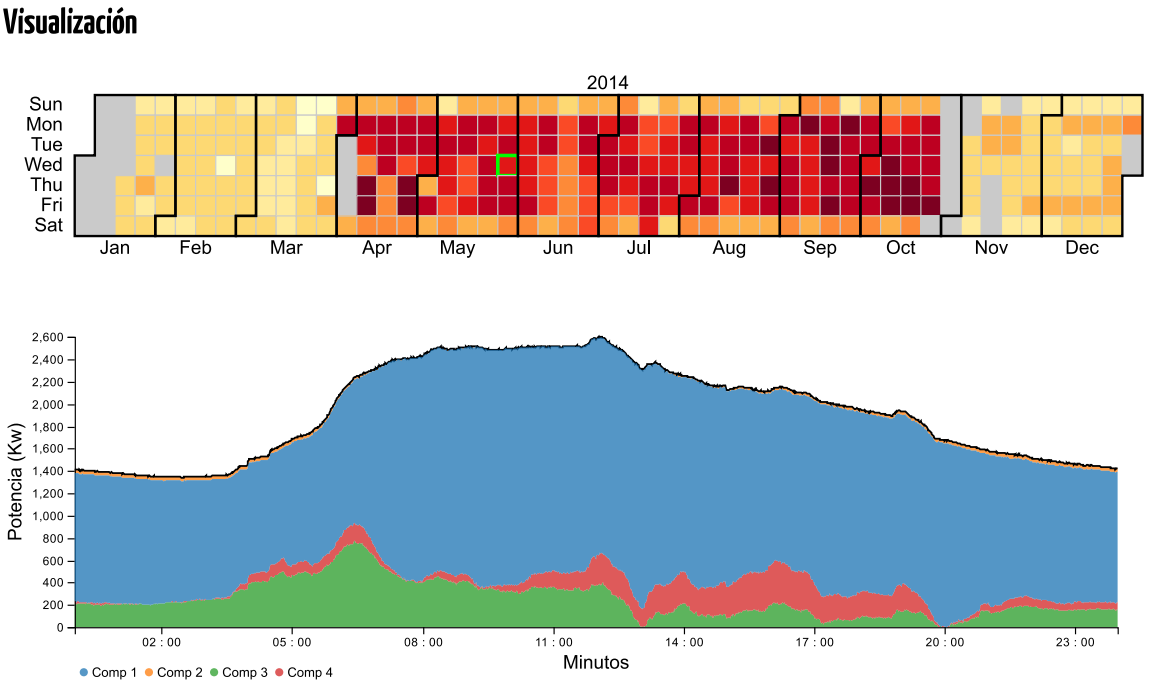

FIGURA 3.9. Visualizaciones asociadas a la componente arranque sistemas HVAC.

3.2.4 Consumo base asociado a grandes consumos a partir del mediodía

La última componente obtenida según la configuración de la figura 3.6, presenta los resultados mostrados en la figura 3.10. En esta componente su influencia es importante solo en algunos días de primavera y verano y especialmente el 9 de abril. Durante este día, personal técnico del hospital hizo pruebas a las máquinas refrigeradoras a partir del mediodía arrancando por fases todas las máquinas. Esto permite asociar esta componente con días de alto consumo y generalmente después del mediodía. Este comportamiento se ve reflejado en el gráfico stacked, haciendo que la joroba típica que muestra el perfil de consumo diario se alargue hasta finales de la tarde.

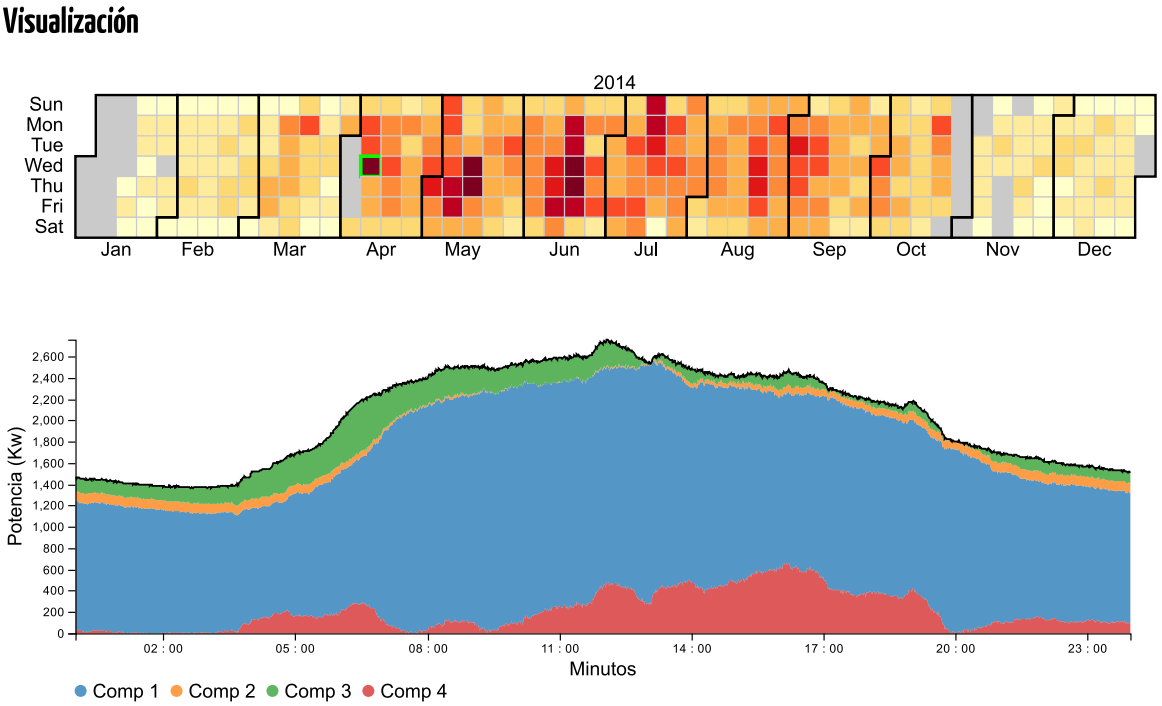

FIGURA 3.10. Visualizaciones asociadas a grandes consumos a partir del mediodía.

3.2.5 Consumo base a baja demanda de frío

Si cambiamos la configuración del algoritmo a una similar a la mostrada en la figura 3.11, una de las componentes resultantes más característica es la mostrada en detalle en la figura 3.12. Al igual que la componente asociada al arranque de los sistemas HVAC, esta componente presenta un comportamiento estacional, pero esta vez en los meses de invierno donde la demanda de frío en el anillo refrigerante es menor.

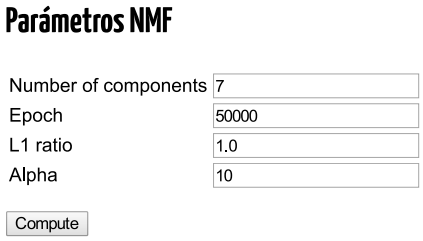

FIGURA 3.11. Configuración del algoritmo para el último caso de estudio.

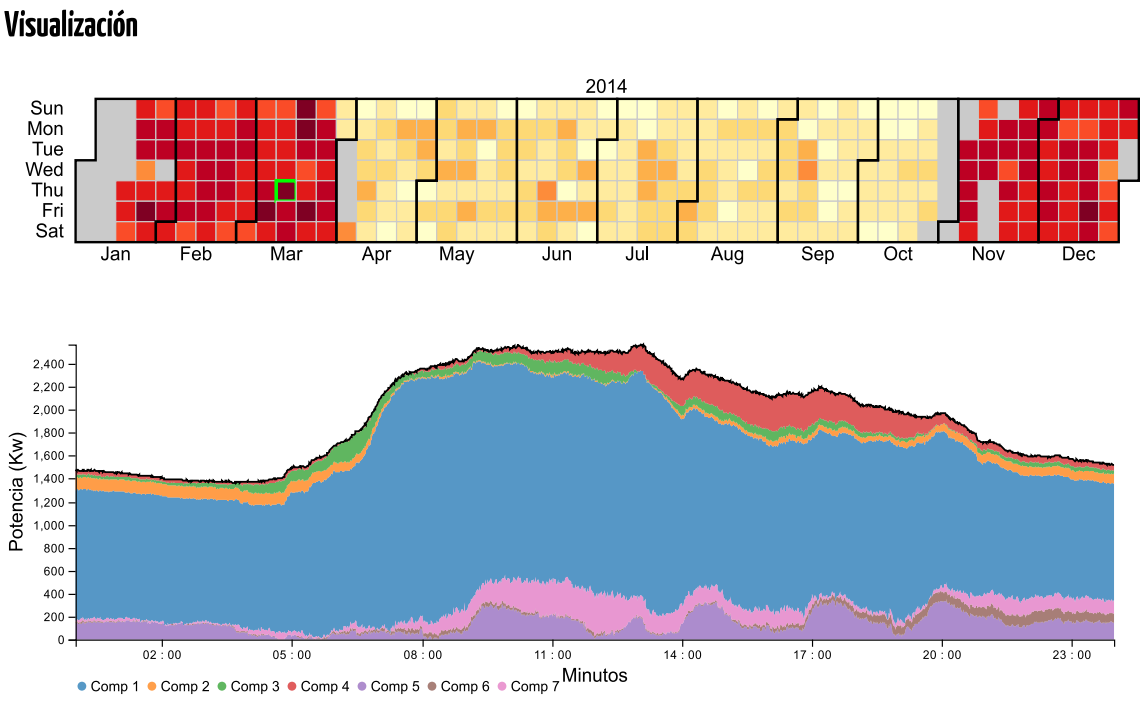

FIGURA 3.12. Visualizaciones asociadas a escasa demanda de frío.

En estos meses de las siete máquinas que pueden generar agua fría al anillo, suele estar activa solamente una de ellas. Estas máquinas están compuestas por tres compresores, que arrancan según la demanda de frío. En los días que señala esta componenete, se demanda el frío justo para un compresor a máxima potencia, de forma que ante un mínimo cambio debe activarse un segundo compresor. Esto supone que la máquina entre en un ciclo límite y que los picos de arranque de los compresores reflejan un consumo, sobre el total, lo suficientemente alto como para que el algoritmo lo identifique, resaltando los picos en el gráfico stacked.

CAPÍTULO 4

DICUSIÓN GENERAL

En este capítulo final se planteará como esta aplicación revela patrones de consumo dentro del hospital de León, cómo es capaz de transmitir esos resultados al usuario y cuales son las líneas de trabajo futuras. En base a los resultados planteados en el capítulo anterior se discutirá las aportaciones del trabajo, las conclusiones que se han obtenido y por último las líneas futuras de este trabajo.

4.1 Aportaciones

Las aportaciones que supone el trabajo aquí presentado son las siguientes:

1. Estudio de la literatura en materia de supervisión de la demanda eléctrica en grandes edificios.

2. Revisión del estado del arte en técnicas de desagregación de energía, basadas en análisis inteligente de datos.

3. Herramienta para la recopilación y el filtrado de la información procedente del Hospital de León.

4. Desarrollo de una aplicación web interactiva para el análisis de consumos eléctricos desagregados del Hospital del León.

5. Estudio e valoración de los resultados obtenidos, identificando eventos relevantes en el consumo del Hospital.

6. Recopilación de todo el trabajo realizado en un artículo de conferencia, para el *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (ESANN) [1], el cual se puede leer al completo en el apéndice B.

4.2 Conclusiones

En grandes edificios, como el Hospital de León, se cuenta con una enorme cantidad de datos procedentes de distintas fuentes, los cuales, tras los oportunos estudios, pueden aportar conclusiones relevantes sobre el comportamiento edificio. Recopilar y estructurar toda esa información fue una tarea propuesta al inicio del trabajo como una etapa previa al análisis inteligente de esta información. En el transcurso de esta tarea se ha llevado a cabo un estudio previo del sistema de medida y de la forma que el hospital recopila la ingente cantidad de información medida. Con este conocimiento adquirido y tras haber realizado la herramienta de gestión, se extrae la conclusión de que es una tarea compleja, la cual requiere mucho esfuerzo tanto en recursos como en tiempo. Aunque se ha solventado la problemática planteada de manera satisfactoria, la capa de gestión de datos sigue en continuo desarrollo, ya que quedan cuestiones por mejorar, como por ejemplo, que el desarrollo propuesto, además de ser desarrollado en Python, debe ser usado en Python lo que en ocasiones supone una desventaja para el usuario, restando portabilidad a nuestra herramienta de gestión de la información.

Una vez concluida la etapa de gestión de la información procedente del hospital, los esfuerzos se centraron en la temática fundamental del trabajo, que es el análisis y visualización de datos integrados en una aplicación web. En un inicio la mayoría de los esfuerzos se dedicaron a la exploración de algoritmos NILM no supervisados basados fundamentalmente en ICA y NMF, concluyendo que este último brinda los resultados más prometedores e interpretables. Tras decantarse por técnicas de factorización no negativa y aplicarlas a datos reales del Hospital de León se logran resultados identificables y que corresponden con las expectativas marcadas, pero que son complicados de trasmitir a un usuario que no este familiarizado con este tipo de factorizaciones matriciales. Por tanto, se reafirmó la idea de que estas técnicas son las adecuadas para el propósito, pero deben ser complementadas con técnicas de visualización interactiva para mejorar la percepción de los resultados por parte del usuario. Tras esta necesidad se integró el análisis en una visualización web interactiva basada en MVC con la ayuda de Flask y D3, potenciando enormemente los resultados obtenidos y creando una sinergia entre análisis y visualización capaz de generar un conocimiento intuitivo de la demanda eléctrica del hospital. Los patrones percibidos por el usuario complementan su conocimiento previo del sistema, dotándole de una perspectiva más amplia que le permita detectar tanto comportamientos anómalos como posibles cambios en la red para la mejora de la eficiencia.

[1] En el momento de la redacción del presente trabajo fin de master, el artículo ha sido aceptado.

Al margen de los resultados puramente técnicos, el trabajo ha servido al proyectante para reforzar sus conocimientos en metodologías de programación, bases de datos, programación orientada a objetos, manejo de lenguajes como Python o Javascript y por supuesto ha afianzado las bases de futuros trabajos de investigación, tesis doctoral o artículos en materia de visualización y análisis inteligente de datos.

4.3 Líneas futuras de trabajo

Aunque se presenta como una herramienta cerrada con la mayoría de los objetivos completados con mayor o menor éxito, el trabajo inspira a desarrollar ampliaciones y mejoras en todos los aspectos fundamentales del mismo.

Empezando por la gestión de la información recogida en el hospital, se ha planteado en la sección anterior que el desarrollo presentado era un desarrollo cerrado en Python, que usa como motor para las bases de datos MS-SQL. Una de las alternativas que se manejan para el futuro es la migración de la aplicación a otro tipo de motores más eficientes como *sqlite* [5], *mongodb* [4] o *elasticsearch* [1]. Además, para la mejora de la portabilidad de la herramienta, una buena tarea sería integrar esta capa intermedia de gestión de datos, al igual que se ha hecho con el análisis y la visualización, dentro de un modelo MVC soportado por Flask u otro framework similar. Esta integración permite, por un lado, extraer los datos en diferentes formatos sin la necesidad de conocer el entorno Python y, por otro lado, fusionar las tareas de gestión, análisis y visualización, ofreciendo al usuario la posibilidad de elegir qué datos tratar, cómo debe ser el análisis y de qué forma quiere que se muestren.

Centrándose en el análisis, queda para trabajos de investigación futuros indagar dentro de la literatura en búsqueda de nuevas implementaciones de algoritmos basados en NMF. Dentro de estás variantes, existen algoritmos que introducen: nuevas restricciones al modelo básico de NMF, modificaciones en la factorización básica para construir modelos convolutivos o que hacen factorizaciones tensoriales *non-negative tensor factorization*(NTF). Todos estas variantes, gracias a la flexibilidad de la aplicación, pueden ser introducidas en la etapa “Compute” de nuestro modelo sin que afecte al resto del comportamiento de la herramienta, agilizando el periodo de evaluación de los resultados gracias a la visualización.

Por último, dentro del campo de la visualización pueden añadirse técnicas visuales de exploración de datos basadas en el concepto *data cube* [48], por medio de librerías Javascript como por ejemplo *crossfilter*. Con este tipo de herramientas se pretende descubrir las correlaciones existentes entre los distintos consumos base obtenidos por NMF, entre sí, y con el resto de variables medidas.

APÉNDICE A

CRONOGRAMA DEL TRABAJO

En el siguiente cronograma se presenta los tiempos orientativos dedicados a las distintas tareas del trabajo:

Tareas	2016/2017																			
	Septiembre				Octubre				Noviembre				Diciembre				Enero			
	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4
INCICIO																				
Documentación y testeo de Python Pandas																				
Documentación sobre aspectos de medida eléctrica																				
Documentación y testeo de Flask																				
Documentación y testeo de SQLAlquemy																				
Documentación sobre algoritmos de desagregación de energía																				
Esquema del trabajo																				
Objetivos																				
DESARROLLO																				
Desarrollo de la aplicación de gestión																				
Desarrollo e implementación del algoritmo NMF																				
Pruebas con distintos parámetros en el algoritmo																				
Desarrollo del artículo de conferencia																				
Integración en flask y d3 de los resultados																				
Optimización de la aplicación																				
DOCUMENTACIÓN																				
Redacción del borrador																				
Revisiones continuas del borrador																				

FIGURA A.1. Cronograma del proyecto.

APÉNDICE B

ATÍCULO SIMPOSIO

Se incluye el texto enviado al *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*.

Latent variable analysis in hospital electric power demand using non-negative matrix factorization

Diego García[1], Ignacio Díaz[1], Daniel Pérez[1], Abel A. Cuadrado [1], Manuel Domínguez [2] *

1- Electrical Engineering Dept. University of Oviedo
Edif. Dept., Campus de Viesques s/n 33204, Gijón - SPAIN
2- SUPPRESS Research Group, University of Leon,
Escuela de Ingenierías, Campus de Vegazana, 24007 León, Spain

Abstract. Energy disaggregation techniques have recently attracted much interest, since they allow to obtain latent patterns from power demand data in buildings, revealing useful information to the user. Unsupervised methods are specially attractive, since they do not require labeled datasets. Particularly, *non-negative matrix factorization* (NMF) methods allow to decompose a single power demand measurement over a certain time period into a set of components or "parts" that are sparse, non-negative and sum up the original measured quantity. Such components reveal hidden temporal patterns and events along this period, related to scheduling events and/or demand patterns from subsystems in the network, that are very useful within an energy efficiency context. In this paper we use this approach on demand data from a hospital during a one-year period, using a calendar visualization of the components, revealing relevant facts about the energy expenditure.

1 Introduction

One interesting approach for power demand analysis [1] is *energy disaggregation*, which decomposes an aggregated electrical measurement into several components, revealing features of downstream demand. An example of this can be found in a household, where we are interested in obtaining different appliance consumptions from the overall energy measurement of the house. In recent years, the interest in energy disaggregation or also named *non-intrusive load monitoring* (NILM) has grown, since the perception and knowledge about consumption is improved through parts-based representation of total energy. This kind of representation can also suggest changes in schedules or detect faults in the network. NILM techniques can be classified into supervised and unsupervised methods. In supervised methods, pattern recognition and optimization algorithms [2] are applied, and its principal disadvantage is that a labeled dataset is required. In many cases, obtaining labeled data can increase set-up costs of NILM systems, making unsupervised methods more suitable. Within unsupervised NILM techniques, *blind source separation* (BSS) and *factorial hidden Markov models*

*The authors would like to thank financial support from the Spanish Ministry of Economy (MINECO)and FEDER funds from the EU under grant DPI2015-69891-C2-1/2-R.

(FHMM) are the most typical approaches as it is explained in [3]. In this paper we focus in BSS methods, specifically in *non-negative matrix factorization* (NMF). NMF, as PCA or ICA, is a matrix factorization with certain constraints. Its principal constraint is the non-negativity in the matrix values, while the PCA and ICA constraints are orthogonality and statistical independence, respectively. Electric consumption measurements are always positive, hence NMF is a suitable decomposition in order to obtain parts-based representation of data as it is shown in [4].

In a real case of electrical consumption in buildings it is well-known that *thermal comfort systems* are almost half of the overall consumption [5]. Thus, if temporal patterns associated with this kind of systems can be obtained, visual tools can be very useful in order to optimize timetables or make changes in certain behaviours to reduce the total consumption. The visualization of NMF components of electrical time series is suggested in our study in order to highlight significant events in the network which can be associated with *thermal comfort systems*.

The remainder of this paper is organized as follows: first, in section 2, we explain basis and limitations of NMF applied to electric measurements. Then, in section 3, results of the application of NMF to hospital power consumption data are presented. Finally, in section 4 conclusions are drawn and some limitations and future works are discussed.

2 Methods

NMF is formulated [6, 4] as an approximate matrix factorization of non-negative input expressed as $\mathbf{V} \approx \mathbf{WH}$. Let us define a M-dimensional vector $\mathbf{v}_i$ whose elements are non-negative and the matrix $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N] \in \mathbb{R}^{M \times N}$ as N observations of $\mathbf{v}_j, j = 1...N$. Considering $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_L] \in \mathbb{R}_+^{M \times L}$ and $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N] \in \mathbb{R}_+^{L \times N}$, NMF can be reformulated as:

$$\mathbf{v}_j = \sum_{\alpha=1}^{L} \mathbf{w}_\alpha h_{\alpha,j}$$

where $\mathbf{v}_j$ is the j-th column of $\mathbf{V}$ and $h_{\alpha,j}$ is the α-th element in j-th column of $\mathbf{H}$. Therefore $\mathbf{v}_j$ is a linear combination of columns in $\mathbf{W}$. The coefficients of the combination are imposed by the elements of columns in $\mathbf{H}$, thus, all the input observations are weighted sums of non-negative basis. This non-negativity suggests that the columns of $\mathbf{W}$ are "the parts of a whole" as it is shown in [7], and they will be referred as *basis consumptions* or simply *components*. Therefore, the elements of the rows in $\mathbf{H}$ contain information about the influence of the basis consumptions along the N observations. The application of NMF to electrical consumptions aims to extract a certain number of basis consumptions that are a parts-based representation of the whole measurement. Basis consumptions show latent patterns, which may contain useful information in order to improve the knowledge about the system.

One difficult issue is to determine the number L of basis components that achieve the exact decomposition $\mathbf{V} = \mathbf{WH}$. Computational NP-hardness of the estimation of L is introduced in [8]. Due to the difficulty of finding L, we dismiss the exact model [8] and we use the basic NMF under only the non-negativity constraints. Typically, $\mathbf{W}$ and $\mathbf{H}$ are estimated using a constrained optimization problem whose objective function is a similarity measurement between $\mathbf{V}$ and $\mathbf{WH}$ [7] and it is solved by means of gradient descent methods [9]. This optimization problem is not convex with respect to $\mathbf{W}$ and $\mathbf{H}$ at the same time; however, it is separately convex in either $\mathbf{W}$ or $\mathbf{H}$. As a nonconvex gradient-based problem, results of basic NMF rely on the initialization of $\mathbf{W}$ and $\mathbf{H}$ matrices. We have considered *nonnegative double singular value decomposition* (*nndsv*) [10].

One variant of the basic method, introduced in [11], is *sparse* NMF, that results in components that are easier interpret, since only a few elements of $\mathbf{W}$ and $\mathbf{H}$ are significant and the rest are near zero (sparseness). Although basic NMF in some cases reach sparse solutions, sparse NMF variants include new constraints in the objective function, so that sparsity and parts-based representation are enforced. The new optimization problem is formulated as:

$$\arg\min_{\mathbf{W},\mathbf{H}} \frac{1}{2}\|\mathbf{V}-\mathbf{WH}\|_F^2 + \alpha\rho\|\mathbf{W}\|_1 + \alpha\rho\|\mathbf{H}\|_1 + \frac{\alpha(1-\rho)}{2}\|\mathbf{W}\|_F^2 + \frac{\alpha(1-\rho)}{2}\|\mathbf{H}\|_F^2$$

where $\|\mathbf{V}-\mathbf{WH}\|_F^2$ is a similarity measurement between $\mathbf{W}$ and $\mathbf{H}$. The rest of the terms in the loss function are elements of regularization. $\|\mathbf{W}\|_1, \|\mathbf{H}\|_1$ are L_1-norm of $\mathbf{W}$ and $\mathbf{H}$ respectively and $\|\mathbf{W}\|_F^2, \|\mathbf{H}\|_F^2$ are L_2-norm. Such regularization terms are weighted by two parameters: α, that affects the whole regularization; and ρ, that controls L_1 or L_2-norm priority. When ρ is 1, only L_1-norm members are significant in the loss function and more sparse results are obtained. On the contrary, if ρ is 0, the loss function is only affected by L_2-norm members and therefore dense basic components are obtained.

2.1 Structure of matrix V

If we consider electrical consumption data as a time series $\{x(t)\}$, a restructuring is needed in order to construct $\mathbf{V}$ as an input matrix to NMF. Thus we can restructure the sequence by breaking it up into N contiguous windows of length M:

$$\{\underbrace{x(0), x(1), \ldots, x(M-1)}_{\mathbf{v}_1}, \ldots, \underbrace{x((N-1)M), x((N-1)M+1), \ldots, x(NM-1)}_{\mathbf{v}_N}\}$$

The new windows conform the matrix $\mathbf{V}$ of observations as follows:

$$\mathbf{V} = \begin{pmatrix} x(0) & x(M) & \ldots & x((N-1)M) \\ x(1) & x(M+1) & \ldots & x((N-1)M+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(M-1) & x(2M-1) & \ldots & x(NM-1) \end{pmatrix}$$

The results of NMF are closely associated to the length of the windows, so a careful choice of this parameter should be done to obtain more interpretable basis components. Power demand time series in buildings typically present a strong cyclic structure for periods of days, weeks, months or years. If the size of the window is chosen according to these periods of time, more interpretable results will be obtained, since the structuring of the results allows a better association between basis components of consumption and events in the network. In other words, different patterns in the basis components are obtained by NMF if a column of **V** is chosen as a one day, week or month. Accordingly, NMF will return components with patterns in daily, weekly or monthly scale.

3 Experimental Results

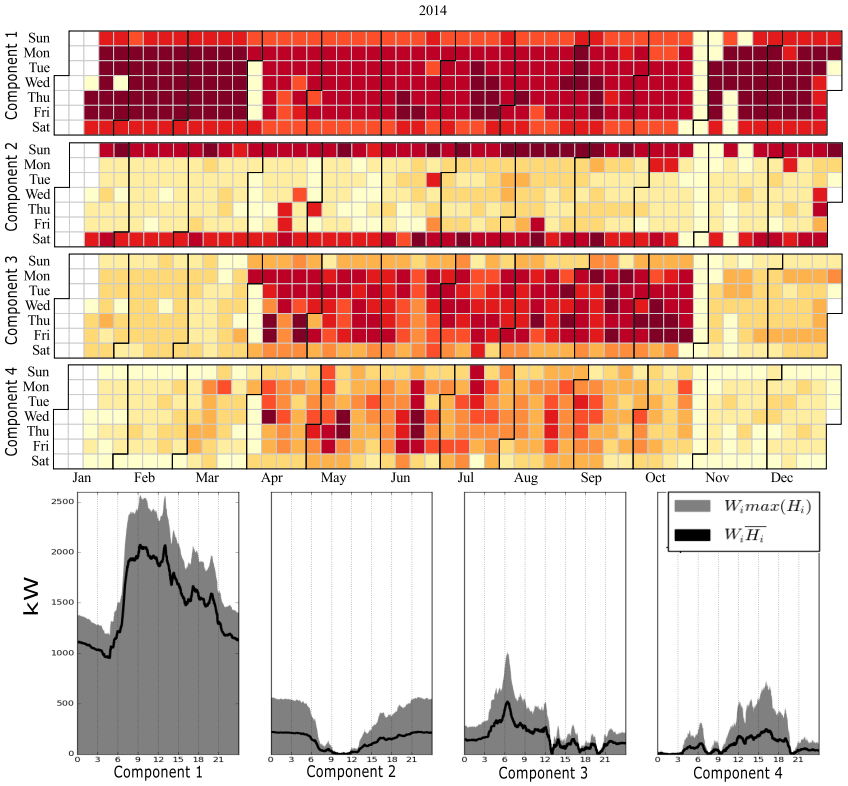

Fig. 1: Four components and their relevance during one year. Above, the calendar visualizations show the component weights (**H** rows) along one year; below, their corresponding maximum and average hourly power demand profiles.

The concepts explained above were tested with real electric power demand data. We have decided to apply sparse NMF to an electrical dataset obtained

from a hospital complex, which includes different measurements of current, voltage, power and quality in several measuring points over the network, collected with a sampling period of one minute. In our analysis, we focus on the total power demand of the hospital, including all the heating, ventilation and air conditioning systems (HVAC), and two cold machines, which feed with water the cold ring that keeps diagnostic devices cool. These two machines have a large peak of consumption in their start. Power data were obtained along 2014 but some days were lost due to problems in the meter caused by a temporary network malfunction.

We choose $M = 1440$ so that one column of $\mathbf{V}$ corresponds to one day; therefore matrix $\mathbf{V}$ will be composed of $N = 365$ observations of daily total power consumption. Regarding the NMF regularization parameter, ρ is set to 1, because we want results as sparse as possible, and α will be greater than 1 in order to increase the influence of L_1-norm in the loss function. Finally, we empirically choose $L = 4$ basic disaggregated components.

The results are shown in Figure 1. Each gray chart shows the product between a basis consumption profile and the highest value in the corresponding row in $\mathbf{H}$, and each black line shows the product between a basis consumption profile and the average value in the corresponding row in $\mathbf{H}$. The calendars show the values of the rows in $\mathbf{H}$ associated with each basis along the year, in other words, the influence of the basis consumption profile. Note that the bottom four charts are in kW units, because the total power demand is a sum of terms, each one being the product of a component and its associated element in $\mathbf{H}$.

The first one has the form of a normal daily profile in the hospital and it is so high that can be interpreted as a canvas to which the rest of the components will be aggregated. The second component shows an important influence in weekends and holidays, where the first one takes less significance. The third component shows a significant influence from April to November and reveals the activation of cooling elements in HVAC systems. If we focus on its profile, we will find a large peak early in the morning at 6:30 when the HVAC system starts. Between November and April, the cooling subsystem in the HVAC system is disabled, and for this reason, the $\mathbf{H}$ elements in the third component are smaller in this period. Finally, the fourth component reveals large values in the afternoon and at noon that are associated with the start of one of the two cold machines that refrigerate the cold water ring. This fact is known because on April 9th the technical staff carried out a maintenance test of these machines.

4 Conclusions

By means of *NMF*, it is possible to find latent patterns in electric power demand time series of large consumers, as shown here with a hospital, which would be more difficult to obtain by other methods. Due to the non-negativity constraint the user can form a daily profile as one weighted addition of positive and characteristic disaggregated profiles where no cancellation between terms is possible. Such lack of cancellations implies that *NMF* can be interpreted as parts-based

representation and these parts reveal connections with relevant events in the network, as, for instance, the start of *HVAC* systems. Moreover, the elements of **H** associated with one of the components contain information about its influence along the N observations. In the hospital results, these elements represent how much relevance the disaggregated profile has along the days in one year. Temporal information and parts-based representation are interpretable and highly useful knowledge, that suggest to create visual analytics applications with the results in *NMF*. We have displayed the weights in **H** matrix using calendar layouts that present a simple representation of the influence of the components in overall daily consumption along a year and allow comparisons between them. In future works *NMF* can be the base of more intuitive and interactive applications. Despite its advantages, some drawbacks are presented by *NMF* techniques applied to electrical consumptions such as the choice of the number of components or the fact that the first component may not be sparse even when sparseness constraints are included. These open issues, as well as exploring other electrical measurements downstream in the network of the hospital, are the guidelines of future works.

References

[1] J Zico Kolter and Matthew J Johnson. Redd: A public data set for energy disaggregation research. In *Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA*, volume 25, pages 59–62. Citeseer, 2011.

[2] George William Hart. Nonintrusive appliance load monitoring. *Proceedings of the IEEE*, 80(12):1870–1891, 1992.

[3] A. Zoha, A. Gluhak, M.A. Imran, and S. Rajasegarar. Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey. *Sensors (Switzerland)*, 12(12):16838–16866, 2012. cited By 127.

[4] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[5] Luis Pérez-Lombard, José Ortiz, and Christine Pout. A review on buildings energy consumption information. *Energy and buildings*, 40(3):394–398, 2008.

[6] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994. cited By 1714.

[7] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[8] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[9] Chih-Jen Lin. Projected gradient methods for nonnegative matrix factorization. *Neural computation*, 19(10):2756–2779, 2007.

[10] Christos Boutsidis and Efstratios Gallopoulos. Svd based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition*, 41(4):1350–1362, 2008.

[11] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

Bibliografía

[1] *elasticsearch web*.
https://www.elastic.co/products/elasticsearchf.
[Online, consultado el de Enero de 2017].

[2] *IEEE SPECTRUM-The 2016 Top Programming Languages*.
https://www.epa.gov/.
[Online, consultado el 06 de Enero de 2017].

[3] *KDnuggets, Languages for analytics*.
http://www.kdnuggets.com/polls/2013/languages-analytics-data-mining-data-science.html.
[Online, consultado el 06 de Enero de 2017].

[4] *mongodb web*.
https://www.mongodb.com/es.
[Online, consultado el de Enero de 2017].

[5] *SQlite web*.
https://www.sqlite.org/.
[Online, consultado el de Enero de 2017].

[6] *Color brewer*.
http://colorbrewer2.org/.
[Online, consultado el 13 de Enero de 2017].

[7] *D3.js website*.
https://d3js.org/.
[Online, consultado el 13 de Enero de 2017].

[8] *Decorators documentation*.
https://www.python.org/dev/peps/pep-0318/.
[Online, consultado el 14 de Enero de 2017].

[9] *EPA web*.

https://www.epa.gov/.
[Online, consultado el 23 de Noviembre de 2016].

[10] *Eurostats web*.
http://ec.europa.eu/eurostat.
[Online, consultado el 22 de Noviembre de 2016].

[11] *Flask website*.
http://flask.pocoo.org/.
[Online, consultado el 13 de Enero de 2017].

[12] *How selections works*.
https://bost.ocks.org/mike/selection/.
[Online, consultado el 14 de Enero de 2017].

[13] *IO tools, Pandas*.
http://pandas.pydata.org/pandas-docs/stable/io.html.
[Online, consultado el 09 de Enero de 2017].

[14] *Jinja website*.
http://jinja.pocoo.org/docs/2.9/.
[Online, consultado el 13 de Enero de 2017].

[15] *Modbus organization web*.
http://www.modbus.org/.
[Online, consultado el 09 de Enero de 2017].

[16] *Pandas web*.
http://pandas.pydata.org/.
[Online, consultado el 06 de Enero de 2017].

[17] *Power Monitoring Expert documentation*.
http://www.schneider-electric.com/en/product-range-presentation/
62919-power-monitoring-expert-8.1.
[Online, consultado el 10 de Enero de 2017].

[18] *Pypyodbc, package website*.
https://pypi.python.org/pypi/pypyodbc.
[Online, consultado el 11 de Enero de 2017].

[19] *Python Tutorial, Python web*.
https://docs.python.org/3/tutorial/index.html.
[Online, consultado el 06 de Enero de 2017].

[20] *Schneider Electric web*.
http://www.schneider-electric.es/es/.
[Online, consultado el 09 de Enero de 2017].

[21] *Scikit Learn web*.
http://scikit-learn.org/stable/.
[Online, consultado el 11 de Enero de 2017].

[22] *SQLAlquemy*.
http://www.sqlalchemy.org/.
[Online, consultado el 06 de Enero de 2017].

[23] *SUPPRESS web*.
http://suppress.unileon.es/es.
[Online, consultado el 09 de Enero de 2017].

[24] *T-SQL documentation*.
https://msdn.microsoft.com/es-es/library/ms188365.aspx.
[Online, consultado el 24 de Enero de 2017].

[25] *Towards Reusable Charts*.
https://bost.ocks.org/mike/chart/.
[Online, consultado el 14 de Enero de 2017].

[26] *Wikipedia, database normalization*.
https://en.wikipedia.org/wiki/Database_normalization.
[Online, consultado el 10 de Enero de 2017].

[27] *Wikipedia, HTML5*.
https://es.wikipedia.org/wiki/HTML5.
[Online, consultado el 13 de Enero de 2017].

[28] *Wikipedia, Javascript*.
https://es.wikipedia.org/wiki/JavaScript.
[Online, consultado el 13 de Enero de 2017].

[29] *Wikipedia, L^p spaces*.
https://en.wikipedia.org/wiki/Lp_space.
[Online, consultado el 11 de Enero de 2017].

[30] *Wikipedia, Weber Law*.
https://en.wikipedia.org/wiki/Weber-Fechner_law.
[Online, consultado el 11 de Enero de 2017].

[31] *wtforms documentation*.
`http://wtforms.readthedocs.io/en/latest/`.
[Online, consultado el 14 de Enero de 2017].

[32] C. BEERI, P. A. BERNSTEIN, AND N. GOODMAN, *A sophisticate's introduction to database normalization theory*, in Proceedings of the Fourth International Conference on Very Large Data Bases - Volume 4, VLDB '78, VLDB Endowment, 1978, pp. 113–124.

[33] C. BOUTSIDIS AND E. GALLOPOULOS, *Svd based initialization: A head start for nonnegative matrix factorization*, Pattern Recognition, 41 (2008), pp. 1350–1362.

[34] M. BURKE, S. M. HSIANG, AND E. MIGUEL, *Global non-linear effect of temperature on economic production*, Nature, (2015).

[35] L. BYRON AND M. WATTENBERG, *Stacked graphs–geometry & aesthetics*, IEEE transactions on visualization and computer graphics, 14 (2008), pp. 1245–1252.

[36] S. J. DAVIS AND K. CALDEIRA, *Consumption-based accounting of co2 emissions*, Proceedings of the National Academy of Sciences, 107 (2010), pp. 5687–5692.

[37] G. W. HART, *Nonintrusive appliance load monitoring*, Proceedings of the IEEE, 80 (1992), pp. 1870–1891.

[38] P. O. HOYER, *Non-negative matrix factorization with sparseness constraints*, Journal of machine learning research, 5 (2004), pp. 1457–1469.

[39] A. HYVAÄRINEN, *Independent Component Analysis A Tutorial*, 1 ed., 1999.

[40] J. Z. KOLTER AND M. J. JOHNSON, *Redd: A public data set for energy disaggregation research*, in Workshop on Data Mining Applications in Sustainability (SIGKDD), San Diego, CA, vol. 25, Citeseer, 2011, pp. 59–62.

[41] D. D. LEE AND H. S. SEUNG, *Learning the parts of objects by non-negative matrix factorization*, Nature, 401 (1999), pp. 788–791.

[42] ——, *Algorithms for non-negative matrix factorization*, in Advances in neural information processing systems, 2001, pp. 556–562.

[43] C.-J. LIN, *Projected gradient methods for nonnegative matrix factorization*, Neural computation, 19 (2007), pp. 2756–2779.

[44] T. MUNZNER, *Visualization Analysis Design*, CRC Press, 1 ed., 2015.

[45] P. PAATERO AND U. TAPPER, *Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values*, Environmetrics, 5 (1994), pp. 111–126.
cited By 1714.

[46] K. PEARSON, *Liii. on lines and planes of closest fit to systems of points in space*, The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2 (1901), pp. 559–572.

[47] L. PÉREZ-LOMBARD, J. ORTIZ, AND C. POUT, *A review on buildings energy consumption information*, Energy and buildings, 40 (2008), pp. 394–398.

[48] S. SARAWAGI, R. AGRAWAL, AND N. MEGIDDO, *Discovery-driven exploration of olap data cubes*, in International Conference on Extending Database Technology, Springer, 1998, pp. 168–182.

[49] E. R. TUFTE AND P. GRAVES-MORRIS, *The visual display of quantitative information*, vol. 2, Graphics press Cheshire, CT, 1983.

[50] S. A. VAVASIS, *On the complexity of nonnegative matrix factorization*, SIAM Journal on Optimization, 20 (2009), pp. 1364–1377.

[51] C. WARE, *Visual thinking: For design*, Morgan Kaufmann, 2010.

[52] R. WIRTH AND J. HIPP, *Crisp-dm: Towards a standard process model for data mining*, in Proceedings of the 4th international conference on the practical applications of knowledge discovery and data mining, Citeseer, 2000, pp. 29–39.

[53] J. S. YI, Y. AH KANG, J. STASKO, AND J. JACKO, *Toward a deeper understanding of the role of interaction in information visualization*, IEEE transactions on visualization and computer graphics, 13 (2007), pp. 1224–1231.

[54] A. ZOHA, A. GLUHAK, M. IMRAN, AND S. RAJASEGARAR, *Non-intrusive load monitoring approaches for disaggregated energy sensing: A survey*, Sensors (Switzerland), 12 (2012), pp. 16838–16866.
cited By 127.