Estudio de los espacios mentales en el aprendizaje de la programación, aportes desde la didáctica de la informática.

Alejandro Miños Fayad

Maestría en Ciencias Cognitivas

Facultad de Ciencias, Facultad de Ingeniería, Facultad de Psicología

Universidad de la República

Montevideo – Uruguay

Junio de 2022

Estudio de los espacios mentales en el aprendizaje de la programación: aportes desde la didáctica de la informática.

Alejandro Miños Fayad

Tesis de Maestría presentada al Programa de Posgrado en Ciencias Cognitivas, Facultad de Ciencias de la Universidad de la República, como parte de los requisitos necesarios para la obtención del título de Magíster en Ciencias Cognitivas.

Director:
Dr. Juan Valle Lisboa

Montevideo – Uruguay
Junio de 2022

Contenido

Resumen

En esta tesis exploramos los modelos cognitivos construidos por estudiantes en respuesta a un problema de programación de mediano porte para dicha población. Si bien los modelos cognitivos son privados y por lo tanto inaccesibles, podemos indagar acerca de su estructura a partir del discurso de los estudiantes. Esto es, al pedirle a los estudiantes que expliquen de qué manera resolverían el problema planteado, la explicación dada será la expresión linguística de su representación mental. Si bien el pensamiento no es necesariamente igual al lenguaje, ambas actividades cognitivas están fuertemente ligadas. El trabajo se basa entonces en suponer que un modelo coherente y bien estructurado llevará a un discurso con esas características y que recíprocamente un discurso coherente y bien estructurado hereda en gran medida esas propiedades del modelo mental.

En el proceso de construcción de un programa, el estudiante debe planificar una solución al problema planteado, el cual posteriormente dará lugar a un programa que lo resuelva. Esta solución está más relacionada con la resolución de problemas que con la elección de técnicas de programación, las cuales serán usadas una vez se haya planificado la solución. La representación de un problema implica la construcción de un modelo cognitivo coherente, el cual como dijimos debería tener su correlato en la expresión lingüística del mismo, dando lugar a un espacio semántico coherente. Al mismo tiempo la representación debe tener en cuenta los aspectos disciplinares propios del problema a resolver y que le dan sentido, dando así coherencia a la solución. Esta representación no es otra cosa que un modelo cognitivo del problema, el cual se evidencia mediante el lenguaje escrito.

En nuestro trabajo consideramos un problema y solicitamos a los estudiantes que expliquen cómo lo resolverían, para luego pasar a construir el programa. Al mismo tiempo realizamos un conjunto de preguntas sobre conceptos básicos

propios de la programación. Observamos que aquellos estudiantes que construyen mejores programas son en general quienes también construyen modelos cognitivos que representan mejor el problema, al tiempo que los espacios semánticos generados presentan una mayor coherencia que en el caso de los estudiantes que muestran peores desempeños en programación. Verificamos también que las respuestas que dan los estudiantes sobre conceptos básicos de programación no están relacionadas con la construcción efectiva de programas.

Desde la didáctica de informática, el estudio de cómo los estudiantes construyen un programa resulta de interés por la propia incidencia de esta sub área en la informática. Al mismo tiempo resulta relevante conocer los procesos cognitivos asociados a la construcción de un programa, pues los mismos tienen relación con el aprendizaje del estudiante, las estrategias de trabajo docente y las actividades a construir. Por último y considerando la relación entre programación y pensamiento computacional, el presente trabajo permitiría arrojar luz sobre los procesos cognitivos asociados al segundo y que son entendidos como relevantes en el sistema educativo nacional.

Capítulo 1

Introducción y marco teórico

1.1 Aspectos generales

Programar es una actividad de una fuerte carga cognitiva, donde es necesario planificar una serie de actividades orientadas a la resolución de un problema (Liao y Bright, 1991; Brennan y Resnick, 2012) al tiempo que se deben conocer y manipular una serie de estrategias y métodos de trabajo (Robins, Rountree y Rountree, 2003). Para programar también es necesario comprender y hacer uso de la sintaxis asociada a un lenguaje de programación, conjuntamente con la semántica del problema (esto es, comprender qué representa y quiere decir la redacción del problema, aspecto que profundizaremos más adelante). Al mismo tiempo se percibe como una dificultad la ausencia de manipulación sobre el objeto en sí mismo y la necesidad de predecir el funcionamiento del programa (Blackwell, 2002), siendo necesario un proceso de abstracción cognitivo el cual se independiza de un producto claramente identificable y manipulable. La importancia de la sintaxis del lenguaje de programación es tal que si no se comprende la misma no es posible resolver el problema en cuestión; cobrando una mayor relevancia el lenguaje en el cual se programa, el cual debe ser tal que facilite el proceso de aprendizaje de las técnicas de programación. En el proceso de aprendizaje la semántica del problema se relaciona con dos dimensiones: primero comprender el problema en sí mismo y segundo relacionar una serie de sentencias de modo que las mismas tengan el sentido esperado, resolviendo por tanto dicho problema. Programar es una actividad que al menos en algunos

lenguajes se asocia a otras dimensiones cognitivas, como la memoria de trabajo, el razonamiento analógico o la resolución de problemas (Prat, Madhyastha, Mottarella, y Kuo, 2020), en tanto que el aprendizaje del mismo tiene relación con las habilidades de adquisición de una segunda lengua (Madhyastha, Mottarella y Kuo, 2020).

Tanto el manejo de la sintaxis como de la semántica del problema convierten a la programación en una actividad con una alta carga cognitiva, en la cual se conjugan múltiples dimensiones. En efecto, el programador (o un estudiante) debe primero comprender el problema a resolver, el cual se expresa en general en lenguaje natural; posteriormente modelarlo (tanto desde lo cognitivo como desde el uso de técnicas de programación adecuadas) de tal forma que el producto genere una representación adecuada del problema, a la vez que permite su programación (Aho, Hopcroft y Ullman, 1998); por último se debe construir un programa, actividad fuertemente dependiente del lenguaje. Comprender el problema es una actividad relacionada principalmente al lenguaje natural y el conocimiento del dominio o la realidad que se pretende modelar. Construir el programa es usar un lenguaje de programación en particular, dominar su sintaxis y relacionar convenientemente un conjunto de sentencias.

La construcción del modelo genera un puente entre la comprensión del problema y su programación, teniendo esta dimensión dos facetas distintas y complementarias: el modelado cognitivo y el técnico disciplinar. El modelo técnico disciplinar (aplicado exclusivamente a la programación) consiste en la elección de sentencias, variables, tipos de datos u objetos entre otros, que en general implica seleccionar formas de representación adecuadas que permitan la programación en un determinado lenguaje. En cambio, el modelado cognitivo (de aquí en más modelado o modelo) es la representación de las grandes relaciones que se dan entre los elementos constitutivos del problema, los cuales sin dar lugar a un programa, sí definen aspectos y dimensiones asociados a cómo debería construirse el programa. Es así que en el modelado del problema la representación debe ser lo suficientemente cercana a la realidad como para representarla convenientemente, sin programarla, a la vez que debe permitir generar pautas sobre cómo construir el programa. El modelado visto de esta forma establece las grandes líneas de trabajo, relación entre elementos del programa y en general grandes restricciones a ser tenidas en cuenta al momento de construir un programa.

La construcción de modelos (cognitivos), sobre los que volveremos más adelante, es uno de los problemas más importantes de la enseñanza y aprendizaje en particular en el campo de la programación, y se relaciona con el

pensamiento computacional. El pensamiento computacional en el sentido dado por Wing (2006) tiene su mejor exponente en la programación (Compañ-Rosique, Satorre-Cuerda, Llorens-Largo y Molina-Carmona, 2015), en particular en enseñanza media, y aunque este trabajo no se enfocará en su estudio, por la propia relación con la programación entendemos necesario hacer algunas observaciones sobre el mismo.

Programar y enseñar a hacerlo, no se puede reducir a comprender o explicar un conjunto de técnicas o sentencias con las cuales construir un programa (Davies, 1933), sino que implica abstraer los elementos relevantes de la realidad (Aho et al., 1998) y resolver el problema pensando en la eficiencia del proceso; dimensiones propias del pensamiento computacional. Es así que el aprendizaje de la programación se transforma también un medio para el aprendizaje del pensamiento computacional (Resnik et al., 2009).

Identificar si se ha comprendido el dominio a modelar es una actividad, en ocasiones, relativamente simple, siendo suficiente realizar una descripción de la realidad para evaluar si se ha comprendido la misma. Para evaluar la correctitud de la programación es necesario analizar si la misma resuelve o no el problema, lo que implica revisar la eficiencia y eficacia del programa. En el caso del modelo evaluar su correctitud es más difícil, pues dicho modelo (técnico en este caso) depende fuertemente de la población a estudiar y sus conocimientos previos. Mientras que para un estudiante de una carrera informática que domina la programación estructurada imperativa modelar es principalmente seleccionar y definir tipos de datos, para un estudiante de un primer curso de programación orientada a objetos el modelado se relaciona con identificar patrones, elementos a iterar y formas de almacenar la información en memoria. Es así que el propio modelado no puede independizarse de los conocimientos previos del estudiante, pues estos determinan el alcance del mismo.

Dowek (2005) recomienda la enseñanza de la informática en tres escalones: enseñanza de tecnologías primero, construcción de algoritmos en secundaria y de ciencias de la computación en el nivel terciario; en tanto que Tucker. (2003) se centra en los primeros niveles. La puesta en práctica de las recomendaciones difiere según el país, centrándose algunos en la formación en tecnologías informáticas en tanto que otros lo hacen en la programación (Webb et al., 2017). En Uruguay los adolescentes de entre 12 y 16 años, se encuentran cursando los primeros años de la educación post primaria, teniendo como asignatura informática. Esta población no está realizando una carrera informática donde la importancia de la programación (Denning et al. 1989) la convierte en un fin en sí misma, sino que la inclusión de la programación pretende el desarrollo de

competencias de resolución de problemas y pensamiento científico (DGES, 2016; DGETP, s.f.). Para este grupo de alumnos, que poseen conocimientos informáticos limitados, modelar no podrá ser elegir tipos de datos (u objetos, según corresponda al paradigma de programación), ni construir pseudocódigos. Si bien es cierto que los estudiantes pueden identificar sentencias o variables a usar, debemos tener en cuenta que esta población no domina técnicas de modelado técnico disciplinar como las mencionadas y dado que el uso de pseudocódigos no asegura construir programas adecuados, los estudiantes se encuentran con pobres formas de representar la realidad.

Pero existe otra dificultad asociada a la construcción de programas y por tanto la forma de modelar. Puesto que el objetivo de este tipo de curso no es la formación de un profesional de la informática, estudiar programación en general tiene otros objetivos, como ser fomentar la abstracción, la integración con otras áreas del conocimiento, el pensamiento científico o el pensamiento computacional (Dowek, 2005; Scaffidi y Chambers, 2012; Maloney, Peppler, Kafai, Resnick y Rus, 2008; Resnik et al., 2009). Las propias actividades que debe realizar el profesor no se deberían centrar exclusivamente en el aprendizaje de aspectos disciplinares (variables, estructuras iterativas, funciones y procedimientos, etc.), sino que deben permitir abordar dichos aspectos sin descuidar la motivación en la actividad (Fiore y Leymonié, 2007), la significatividad de las mismas y la posibilidad de integración con otras áreas del conocimiento (Antúnez, del Carmen, Imbernon, Parcerisa y Zabala, 1992). De este modo, la dificultad relativa de las actividades hace que las mismas no siempre evidencian elementos fuertes de modelado desde el componente disciplinar. Por lo expuesto se genera una sobrecarga cognitiva en los estudiantes (Astolfi, 1999) dada por la forma de reconocer la adecuación de un modelo e identificar las relaciones lógicas que hacen a la propia solución; sea por el desconocimiento de las técnicas de modelado, sea por las características del problema en sí mismo.

No obstante lo anterior, el modelado existe y está presente, aunque no sea mediante la aplicación de técnicas de programación y se visualice de otras formas. Si entendemos que el lenguaje es un medio de sentir, experimentar y pensar (Chomsky, 1957), al momento de modelar el estudiante debe poder expresar las relaciones fuertes entre los grandes elementos del programa, dando por tanto lineamientos de soluciones posibles al problema planteado. Asumimos que la relación entre el modelo y el programa, más allá de la técnica usada, se debe visualizar en la verbalización de dichas relaciones, sin que esto signifique a su vez una simple traducción de la consigna de trabajo. El modelado así entendido es una representación del problema que ha captado el sujeto, su

entendimiento y cómo cree que debe ser representado o resuelto. De este modo el modelo deberá dar cuenta de dos condiciones simultáneas: ser adecuada desde lo disciplinar según Aho et al. (1998) y constituir un espacio semántico en el sentido dado por (Valle-Lisboa y Mizraji, 2007).

La perspectiva disciplinar, programacional en nuestro caso, no es otra cosa que el análisis y evaluación de la correctitud de las relaciones desde la construcción del programa solicitado. Dado que verbalizar la solución a un problema es organizar y llevar a la práctica un discurso coherente que está constituido por distintos temas relacionados (Valle-Lisboa, Pomi, Cabana, Elvevåg y Mizraji, 2014), la construcción de la solución a un problema de programación implicaría también tomar conciencia de los elementos constitutivos del mismo, expresándolos de forma coherente. Ya que el discurso deber dar cuenta de las ideas principales del problema a resolver, debería evidenciarse una superposición semántica entre la expresión del modelo y la realidad a modelar (Crossley, Kim, Allen y McNamara, 2019); y en este caso en particular de los conceptos fuertes desde la programación de la solución. Esta representación debería considerarse como una autoexplicación de aquello que se ha entendido, debiéndose observar una mayor cohesión y riqueza de vocabulario en las personas que realizan buenas explicaciones que en el resto de los individuos (Allen, Snow y McNamara, 2015).

En este trabajo analizaremos qué características tiene el modelado de programas en estudiantes que están cursando los dos primeros años de educación post primaria. Para lograr lo anterior cotejaremos las resoluciones efectivas a programas propuestos, con las formas de representar el problema, construir un modelo, lo cual se deberá traducir en la representación escrita del mismo. De este modo es que les pediremos a los estudiantes que expliquen cómo resolverían un problema, identificando y evaluando nosotros los elementos que dan lugar a un adecuado modelado; posteriormente evaluaremos la solución que han presentado y que es el resultado de la programación. El análisis de la información será tal que conjugará aspectos disciplinares con las relaciones semánticas que se establecen entre los conceptos usados por los estudiantes, la estructura del discurso y técnicas de clustering. En este análisis se conjugan aspectos disciplinares, propios de la programación y el lenguaje empleado, con el análisis semántico de aquello escrito por los estudiantes.

1.2 Lenguaje natural y representaciones

El lenguaje no es solo una forma de comunicación, de intercambio de información o representación de ideas o hechos, sino que además es un instrumento del pensamiento; en palabras Chomsky (1957) “language is not merely a means of expression and communication; it is an instrument of experiencing, thinking and feeling, as well as a means of self-expression and personal growth” (1957, p. 4). El lenguaje permite al individuo apropiarse de la realidad, ir más allá de la percepción sensorial instantánea, de modo que la estructura cognitiva se refuerza a medida que se hace uso de la capacidad de conocer su entorno (Herrera, 2016). La importancia del lenguaje es tal que, que ya tempranamente Gal'perin (1989) sostenía que en el proceso de aprendizaje, la verbalización de la acción es condición necesaria para que aquello a aprender forme parte de la estructura cognitiva del sujeto; aspecto que incluso al día de hoy es tenido en cuenta en el ámbito educativo y las recomendaciones didácticas. Es por esto que entendemos necesario comprender el problema desde una perspectiva asociada al uso del lenguaje, el cual entendemos permite representar adecuadamente relaciones entre resultados y procesos de aprendizaje o cognitivos.

Tres conceptos deben ser considerados en nuestro trabajo: la semántica, los espacios mentales y los espacios semánticos.

La semántica es el significado lingüístico de un enunciado (Frawley, 2013) , pues ella “tiene que ver con nuestra capacidad de representarnos el mundo mediante símbolos” (Gutierrez, 2006, p. 1). La semántica de un texto no es la sumatoria de las semánticas de cada una de las frases o palabras del mismo, sino que tiene relación con el texto en su conjunto, que también se denomina una macroestructura semántica (Van Dijk, 1980, p. 44) y capta lo que en general se denomina tema. Kintsch y van Dijk. (1978) afirman que la semántica tiene una microestructura, donde el significado está dado por las proposiciones y sus relaciones, y una macroestructura, asociada al discurso como un todo. En la comunicación la semántica tiene dos polos: la intención de quien construye y expresa y el problema por un lado y la interpretación y posterior apropiación y comprensión de dicho problema por parte de quien debe resolverlo.

Las experiencias personales así como nuestra interpretación del mundo dan lugar a lo que se denominan modelos mentales, los que a su vez vuelven a condicionar nuestra experiencia, al tiempo que procuran brindar respuestas coherentes a las experiencias del individuo (Van Dijk, 2005). Un caso particular

de modelos mentales son los modelos de contexto, los cuales se construyen en función de las experiencias comunicativas (Van Dijk, 2004), en particular mediante el lenguaje. Las representaciones que construimos del mundo y que percibimos a través del lenguaje se ubican en lo que se denomina Semántica cognitiva, disciplina que busca "desentrañar la organización oculta… [que ocurre] mediante el lenguaje y el pensamiento" (González, 2005, p. 106). El hecho de leer un texto, o escuchar un discurso, da lugar a una interpretación personal y subjetiva de aquello que es evocado, es decir, conforman un modelo de contexto en el sentido dado por Van Dijk (2004). Por tanto, el individuo hace una interpretación personal de un término, como ser "programa", al cual le asigna propiedades y características en función de su experiencia. Son los denominados espacios mentales esos espacios cognitivos, asociados a conceptos y que tienen sentido en función de la realidad del individuo (Pascual, 2012) , los que permiten comprender y darle coherencia al discurso, que son "reversibles y ni siquiera tienen que ser lógicamente consistentes" (González, 2005, p. 112).

Según Kintsch (1998), el conocimiento se puede visualizar como una red de nodos donde cada uno de ellos representa un concepto. El significado de los nodos no es fijo, sino que depende del contexto en el cual se inserta. Es así que al usarse un concepto, su significado se construye en la memoria de trabajo en función de su contexto, esto es, en función de los otros conceptos asociados al concepto original. De este modo, el significado de una palabra está condicionado por las palabras que se relacionan con la primera. Una forma de relacionar conceptos está dada por la construcción de un vector de conceptos, donde cada posición de un vector tiene asociado un número, el cual a su vez relaciona el concepto original con otro. El autor se pregunta qué representan los números anteriores, respondiéndose que tal vez sea la experiencia de la vida, la interacción con el mundo, siendo por tanto producto de nuestra relación con el entorno. Como forma de analizar las relaciones entre los conceptos, el autor afirma que el análisis semántico latente es una técnica que permite identificar la relación semántica entre palabras, las cuales están insertas en un contexto, existiendo otros algoritmo que también permiten identificar estas relaciones (Altszyler, Sigman, Ribeiro y Slezak, 2016).

En el caso del análisis semántico latente, u otra herramienta como Word2Vec (Mikolov, Chen, Corrado y Dean, 2013a), las relaciones entre los conceptos se modela computacionalmente mediante el uso de vectores, los que de alguna forma permiten representar las relaciones semánticas entre palabras. Es así que los denominados espacios semánticos no son otra cosa que la representación de

las relaciones existentes entre un conjunto de conceptos, que determinan las relaciones semánticas que evidencia el individuo. La verbalización o escritura de la solución a un problema implica poder organizar un discurso coherente (Valle-Lisboa, Pomi, Cabana, Elvevåg y Mizraji, 2014), el cual a su vez debe poder seguir una trayectoria fluida entre los distintos sub temas que conforman la solución (Valle-Lisboa et al., 2014). Dicho de otro modo, es necesario planificar u organizar el discurso, prever la solución o el cumplimiento del objetivo y la relación de lo anterior con el propio lenguaje. Más aún, dicho discurso debería relacionar las principales ideas que dan lugar a la propia solución; las cuales están condicionadas por los espacios semánticos del individuo y generan a su vez su propio espacio semántico. En el caso de un problema computacional como un programa, debería verse la identificación y relación de los conceptos fuertes que dan lugar a la solución. Si bien tanto la verbalización como la escritura podrían ser indicadores del proceso de planificación u organización conceptual, vale mencionar que el habla "es espontánea y tiene muchas propiedades del habla improvisada: pausas, errores, reparaciones… La escritura de textos está, en general, más controlada… Es decir, mientras que el lenguaje oral es "lineal"… [en tanto que la escritura permite la] "composición", de retroceder y reescribir" (Van Dijk, 2019, pp. 23 – 24). Por tanto, podríamos estar afirmando que el uso de la escritura sería un indicador particularmente adecuado para analizar el proceso de planificación del discurso. Por otro lado podemos afirmar que comprender un problema, hecho o parte de la realidad, no es otra cosa que hacer una interpretación del mismo, basado en el conocimiento del problema y su descripción (Johnson-Laird, 2010). Esto sugiere que el estudio de las producciones lingüísticas de los sujetos podría acercarse a obtener información relevante de los procesos cognitivos que subyacen a la resolución de problemas.

Si bien es posible la realización de análisis cualitativos del discurso, como ser en la relación entre palabras o ideas expresadas por el escribiente, existen herramientas basadas en el análisis semántico latente (Landauer y Dumais, 1997) o en Word2Vec (Mikolov, Chen, Corrado y Dean, 2013a), que permiten identificar relaciones dadas en la entrada de datos, construyendo así un espacio semántico (Valle-Lisboa y Mizraji, 2007). Por tanto, este tipo de herramientas es útil para objetivar las relaciones entre palabras, definiendo la semántica en función de su contexto y no del capricho del investigador. En el caso de la explicación sobre cómo resolver un problema, sería posible determinar si dos conceptos tienen relación efectiva en función del contexto (como ser los términos variable y datos, por ejemplo), en lugar de dejarlo librado a la interpretación del revisor.

Es así que se podrían estar usando herramientas de análisis semántico como forma de determinar qué tan adecuada, precisa o correcta es la construcción de una solución, como ser el modelo de un problema. Estos criterios deberían considerar algunos aspectos, como por ejemplo:

1. La superposición de las explicaciones que se deberían dar entre aquello a explicar y cómo es explicado.
2. La identificación de los principales conceptos que son parte del problema y su relación en un discurso coherente.
3. Una mayor capacidad de explicar el problema entre quienes tienen un dominio disciplinar mayor en relación a quienes presentan un menor dominio disciplinar.

A modo de ejemplo, en el estudio de la realización de resúmenes se pueden analizar propiedades del texto producido que se relacionan con el nivel de conocimiento que los sujetos tienen del texto original sobre el cual se realiza el resumen (Crossley et al., 2019). En este tipo de estudios, una de las variables relevadas es la cohesión del texto, la cual según Allen et al. (2015) es la relación entre las grandes ideas del mismo con la cantidad de conectores y superposición de ideas mediante palabras y frases, definiendo un espacio semántico (Valle-Lisboa y Mizraji, 2007) o mental adecuado. Los autores analizan las autoexplicaciones de individuos, concluyendo que entre quienes realizan mejores explicaciones se observa una mayor cohesión de las mismas. De este modo deberíamos estar esperando textos más cohesivos entre quienes realizan mejores modelos del problema a programar que entre tienen peores desempeños.

Un aspecto a considerar en relación a la representación de un problema mediante la escritura lo constituyen los conocimientos previos. MacNamara (2004) presenta un estudio relacionado con la biología, que sostiene puede extenderse a otras áreas científicas, en el cual las personas que tienen un buen conocimiento de dominio son también quienes realizan mejores autoexplicaciones. Este conocimiento del dominio es en sí mismo un espacio semántico, es decir, un conjunto de conceptos relacionados que a su vez en este caso representan convenientemente al realidad.

En el caso de las explicaciones asociadas a cómo resolver un problema computacional, para el cual se entrega previamente una consigna, cabe considerar si el conocimiento del dominio, que está en línea con la consigna, está relacionado con la explicación de cómo resolver el problema. De este modo, una explicación poco adecuada sobre cómo se resuelve el problema podría estar

relacionado con las capacidades de modelado del estudiantes, con el conocimiento del domino de eso a resolver o incluso con el incorrecto uso de los significados de términos. Esta representación que se hacen de la realidad será tal que una mayor sofisticación léxica se asocia a una menor repetición de palabras, lo cual a su vez también se asocia con una mayor cantidad de palabras usadas (Skalicky, Crossley, McNamara y Muldner, 2017).

En relación a los resúmenes es de esperar que aquellos bien construidos presenten una superposición semántica con el texto a resumir (Crossley et al., 2019). Una situación similar a lo antedicho deberíamos estar esperando en el caso que sea necesario representar cómo se debe resolver un problema, donde la representación equivaldría al resumen y el texto al problema en sí mismo. De este modo, las expresiones de soluciones a construir, modelos o representaciones de programas, deberían evidenciar una superposición semántica con la realidad a resolver; superposición que también quedaría en evidencia mediante la referencia a los principales conceptos del problema desde lo computacional.

1.3 Informática, programación y pensamiento computacional

Entendemos la Informática como una disciplina científica (Denning, 2005) la cual se encarga de estudiar qué es efectivamente computable (Denning et al. 1989), con tradiciones tanto empíricas como formales (Dodig-Crnkovic, 2002), las cuales condicionan la construcción de conocimiento. La Informática tiene como área constitutiva y fundamental la programación, la cual afecta directa o indirectamente a la casi totalidad de las sub áreas de la misma (Denning et al., 1989), siendo por tanto un fin y un medio de estudio a la vez. La programación es a su vez el estudio de las técnicas que permiten construir programas. Si bien existen visiones distintas de la naturaleza ontológica de los programas de computación (Nicola, Primiero y Turner, 2021), en el presente trabajo no abordaremos dicha discusión. Una programa es una conjunto de instrucciones, en un determinado lenguaje de programación, que permiten controlar el procesador de un ordenador digital, de modo que todos los programas se asocian a un algoritmo (van Emden, 2015). El estudio de la programación y las formas asociadas a su enseñanza y aprendizaje, cobran una vital importancia no solo por la manipulación que se hace del hardware, sino pues cómo y qué aprenden los estudiantes, así como las formas de trabajo de los docentes,

permitirían fortalecer los procesos de enseñanza y aprendizaje de la propia programación y por tanto de la informática.

Una de las dimensiones a estudiar en relación a la programación es la elección del primer lenguaje, sobre el cual existe una falta de consenso en el sentido de determinar cuál es el mejor de ellos para un primer curso de programación (Koulouri, Lauria y Macredie, 2014). Wiedenbeck, Ramalingam, Sarasamma y Corritore (1999) sostienen que en el caso de los lenguajes orientados a objetos su uso en un primer curso de programación no estaría asociada a mayor facilidad, pues existiría una dificultad significativa en comprender el programa como un todo (que en general son de tamaño relativamente grande), sus funcionalidades y el flujo de control. Ehlert y Schulte (2009) sostienen que cuándo comenzar con el paradigma objetos no parece ser la principal dificultad en un primer curso de programación, centrándose en la secuencia de contenidos y su abordaje. Sin embargo, los autores observan que cuando un curso comienza con el abordaje de la orientación a objetos, temas como el manejo de iteraciones, estructuras de control y estructuras datos, resultan más difíciles que en el caso de postergar el trabajo con objetos.

La pertinencia o no de un determinado lenguaje y paradigma sigue estando presente en la discusión actual. Fernández, Peña, Nava y Velázquez (2002) analizan la elección del primer lenguaje de programación y su relevancia al momento de definir el alcance de los saberes a estudiar. Mientras el uso de un lenguaje estructurado permite aprender los conceptos básicos de la programación (iteraciones y estructuras de control), tiende a postergar el modelado del problema (como el encapsulamiento y la modularización), esto es, identificar los grandes elementos constitutivos que dan lugar a la solución; aspectos identificados por Wiedenbeck y Ramalingam (1999). Esto último evidencia la tensión existente entre centrarse en la implementación y la visión general del problema, tensión que se traslada a los cursos de programación, en particular a los primeros. Otro enfoque posible es el abordaje de la programación desde el paradigma de la orientación a objetos, el cual en ocasiones se argumenta presenta una mayor simplicidad de modelado. No obstante lo anterior, la bibliografía evidencia algunos problemas al comenzar programando con este paradigma, como ser: postergación del aprendizaje de las técnicas básicas de programación (lo que se conoce como posponer la programación) o la tendencia a centrarse en la presentación e interfaz en detrimento de la algoritmia básica. Lo expuesto se complejiza por el denominado “problema del desplazamiento” (Fernández et al., 2002), es decir, el tiempo necesario para aprender un paradigma luego de trabajar con otro; tiempo que en un programador experto lleva un mínimo de seis meses.

La otra dimensión a considerar es el lenguaje a usar como herramienta para el aprendizaje de la programación. Mientras un mismo lenguaje de programación puede usarse para enseñar varios paradigmas, al tiempo que reduce las dificultades asociadas al aprendizaje de la sintaxis, no tiene en cuenta el hecho que el lenguaje se relaciona a un paradigma y a la forma de pensar una solución en particular (Fernández et al., 2002). Existe una tensión y dificultad evidente entre la forma de pensar la solución, el problema y el aprendizaje de la sintaxis del lenguaje a usar, lo cual se trasluce en la preocupación por elegir el mejor lenguaje de programación que permita centrarse en el aprendizaje de las técnicas de programación por sobre las particularidades del mismo (Grandell, Peltomäki, Back y Salakoski, 2006). En tal sentido Koulouri, Lauria y Macredie (2014) sostienen que existe un impacto significativo en la elección del primer lenguaje de programación, destacando las fortalezas que evidencia un lenguaje sintácticamente simple como Python.

Ahora bien, más allá del hecho de programar aprendiendo métodos y técnicas que lo permitan, el aprendizaje de la programación se relaciona con pensamiento computacional en el sentido dado por Wing (2006). Aunque existen variaciones en la definición de qué es el pensamiento computacional, existe una suerte de consenso en que el mismo implica hacer uso de la descomposición y abstracción como formas de trabajo (Shute, Sun y Asbell-Clarke, 2017), al tiempo que Wing (2010) sostiene que el mismo es el “the thought processes involved in formulating problems and their solutions” (p. 1). Mientras la descomposición no es otra cosa que dividir el problema en otros de menor tamaño o dificultad, la abstracción es la eliminación de detalles innecesarios del modelo, de modo que aquellos que quedan son relevantes para la representación de la realidad (Wing, 2006). La diferencia entre programación y pensamiento computacional es sutil, pues aunque la segunda no requiere de la primera, la mejor forma de operativizar el pensamiento computacional es a través de la programación (Webb et al., 2017). Pero el aprendizaje de habilidades de programación no solo permite el desarrollo del pensamiento computacional (Resnik et al., 2009), sino que además tiene efectos cognitivos positivos en áreas como la planificación, resolución de problemas a través de computadoras y razonamiento lógico entre otros aspectos (Liao y Bright, 1991), que deberían ser considerados.

Es así que la enseñanza de la programación puede tener distintos objetivos dependiendo si es tomada como un fin o medio. La programación en carreras informáticas (o como asignatura complementaria en carreras no informáticas) será una forma de manipular técnicas básicas de la programación que son relevantes para la informática, pues estructuran el currículo (Denning et al., 1989). En cambio, la enseñanza de la programación en enseñanza media estará

orientada a procesos de alfabetización científica o al desarrollo del pensamiento computacional Wing (2006). En ambos casos existen dos beneficios en el aprendizaje de la programación que no pueden dejarse pasar por alto. Programar implica construir representaciones del problema a modelar (Aho et al., 1998) a la vez que “reconocer la diferencia que hay entre hacer que un programa funcione y lograr que lo haga bien” (Jackson en Pressman, 2010, p. 189). Dicho de otro modo, programar está asociado no solo a resolver problemas codificando en un lenguaje de programación una solución, programar implica comprender y adquirir la disciplina necesaria para abstraer un problema y resolverlo de forma eficiente.

1.4 Modelos, programación y cognición

Hasta ahora hemos usado el término modelo de forma general, correspondiendo por tanto ser más preciso en sus características. ¿Qué es un modelo? Un modelo es una representación del problema a resolver, en el cual se relacionan los elementos constitutivos del mismo, estando por tanto condicionado por el tamaño de dicho problema y los saberes de quien modela. El modelo de un problema es una etapa asociada a la construcción de un programa que se encuentra entre la comprensión de la consigna y el modelado técnico disciplinar. Crear un modelo es comprender el problema, identificar las grandes relaciones entre aquellos elementos que a su vez se entiende determinan el problema. Esta etapa será la que permitirá posteriormente crear un modelo técnico disciplinar.

Como dijimos, el modelo está condicionado por el tamaño del problema y la persona que modela. En efecto, en un proyecto de gran porte, el modelo podrá estar en función de la estructura de componentes, paquetes y en general con las distintas vistas del sistema en el sentido dado por Kruchten (1995). Si bien la representación anterior es disciplinar, también es cierto que un experto podrá entender que esa forma es adecuada para identificar y representar las grandes relaciones evidenciadas, cumpliendo los dos roles a la vez. En cambio, cuando un individuo está aprendiendo las bases de la programación estructurada imperativa, y si por ejemplo se le solicita construir un programa que dado un número determine si es positivo, el modelo se podría relacionar con la definición de variables o condiciones. Es por esto que las características que tiene el modelo están condicionadas por el tamaño del problema en sí mismo. En un problema de tamaño relativamente grande, el modelo se distancia significativamente del algoritmo y se centra en las relaciones entre elementos.

En cambio, cuando el tamaño del problema es reducido, el modelo es esencialmente el algoritmo, pudiendo incluso a discutirse si existe como tal.

Aunque dijimos que un modelo no es un algoritmo, debemos hacer la aclaración que eso implica que las propias relaciones evidenciadas en el modelo se pueden considerar como un algoritmo, pero más genérico. Dicho de otro modo, un caso de uso en el sentido dado por Pressman (2010) no es un algoritmo con un alto grado de detalle, pero sí es una secuencia de pasos que puede dar lugar a un posterior algoritmo computacional. Como adelantamos previamente, esta percepción de modelo debe diferenciarse del modelo técnico disciplinar, donde lo que se busca es aplicar técnicas de modelado o programación que permitan resolver un problema. Así, un modelo no es la elección de tipos de datos u objetos o un algoritmo, o incluso la descripción precisa de cuándo se usaría una estructura de repetición, una función o procedimiento; pues como dijimos sería un modelo disciplinar, computacional. En nuestro trabajo un modelo es cognitivo. Aunque un modelo puede coincidir con el análisis y descripción de un algoritmo, para problemas de reducido tamaño, cuando esto ocurre el propio tamaño del problema se convierte en una limitante, pues se confunde la identificación y relación de los grandes elementos del problema con su resolución efectiva. Es por lo antedicho que entendemos, y así será considerado en este trabajo, que solo es posible identificar un modelo cuando el mismo se “aleja” del algoritmo y no es representado como este último.

Para limitar o informar de un modelo es necesario expresarlo de forma adecuada, en nuestro trabajo y siguiendo lo previamente expuesto, mediante el lenguaje escrito. Dado que el modelo debe identificar aquellos elementos más importantes del problema, relacionarlos y a la vez construir una enunciado que de cuenta de las relaciones existentes, es que el modelo se convierte en un espacio semántico, el cual relaciona de forma fluida los elementos constitutivos del problema (Valle-Lisboa et al., 2014) y se encuentra limitado por los modelos mentales en el sentido dado por Van Dijk (2005).

Debe tenerse en cuenta que no existe un modelo intrínsecamente correcto o incorrecto, sino que el mismo es relativo a cada persona, sus estructuras cognitivas y los resultados que se desprenden de la construcción de la solución efectiva. Consideramos el caso de la construcción de un programa, el cual es correcto si resuelve lo solicitado, en otro caso está mal. Sin embargo, desde la evaluación que hace una persona (por ejemplo un docente) sobre el programa, el mismo puede no resolver completamente el problema e igualmente estar en un nivel de aceptablidad. Claramente es el docente quien determina los grados

de libertad que tiene la solución, qué tan relevante es una omisión o cómo afecta en la lógica del problema. Al igual que la evaluación, lo mismo ocurre con los modelos, los mismos son correctos en función de un contexto.

Varias dimensiones cognitivas se relacionan en la construcción de un modelo, algunas asociadas a actividades de planificación, otras con el lenguaje, por ejemplo. A continuación presentamos una serie de trabajos en los cuales se evidencia la relación entre la programación y dimensiones cognitivas.

En el caso del uso del lenguaje Logo, Clements y Gullo (1984) observan que los estudiantes tienden a producir ideas originales, incrementar la habilidad para la resolución de problemas, e incluso podría estar afectando positivamente la capacidad de automonitoreo del propio aprendizaje. En el caso de las habilidades de resolución de problemas o planificación, se observa una frecuente referencia a los beneficios que aporta la programación, entre otros los trabajos de Liao y Bright (1991) y Brennan y Resnick (2012).

La programación es una actividad exigente no solo por el hecho de ser necesario planificar y proponer efectivamente la solución a un problema, sino porque es necesario dominar varias estrategias de resolución de problemas. En tal sentido Robins et al. (2003) afirman que una de las diferencias entre los programadores expertos de los nóveles es conocer (y por tanto recordar) y aplicar una serie de estrategias y métodos de trabajo asociados a la resolución del problema planteado. Sin embargo, conocer o describir estrategias no es suficiente para programar, por lo que Davies (1933) diferencia entre comprender y aplicar estrategias de programación. Mientras las primeras son de tipo declarativo y se centran en describir y comprender los enunciados o problemas, las segundas los ponen en práctica, debiendo por tanto incluir a las primeras.

Como afirma Blackwell (2002), programar implica la pérdida de la manipulación directa sobre el objeto, siendo necesario un proceso de abstracción que es mediado por alguna notación (la propia del lenguaje), la cual a su vez determina otras tantas notaciones (es decir, usar un conjunto de sentencias y tipos de datos en función las definidas previamente). Es así que la programación estaría relacionada con capacidades de abstracción no solamente en relación a la construcción de soluciones mediante artefactos y construcciones simbólicas, sino que además por el hecho de prever el funcionamiento de estas construcción antes de su propia ejecución.

Al mismo tiempo se observa también la incidencia positiva de la programación en las habilidades de uso del lenguaje (Maloney et al., 2008), así como en el

fortalecimiento de las comunicaciones interpersonales (Scaffidi y Chambers, 2012). La comunicación y el lenguaje en general parecen ser variables importantes y que están relacionadas con la programación y su aprendizaje. En efecto, se ha observado que las personas que no tienen el inglés como idioma nativo tienen mayores dificultades para comprender programas (Guo, 2018). Al mismo tiempo, para una correcta programación es necesario relacionar piezas más grandes de software, a la vez que se entienden y comentan.

Codificar implica una fuerte relación entre los conocimientos de los lenguajes informáticos y el lenguaje natural, donde en ambos casos es necesario integrar la sintaxis con la semántica, reconocer palabras, relacionar sentencias o párrafos o inferir consecuencias entre otros aspectos (Fedorenko, Ivanova, Dhamala y Bers, 2019); relación que correspondería analizar en función de la adquisición de una segunda lengua. En el caso de Python, se ha observado que las habilidades aritméticas no son el mejor predictor de las de programación, destacando el razonamiento analógico, resolución de problemas y la memoria de trabajo (Prat, et al., 2020). Más aún, la relación vista por Fedorenko et al. (2019) tiene relación con lo expresado por Madhyastha, Mottarella, y Kuo (2020), quienes observan que la tasa de aprendizaje del lenguaje está fuertemente determinada por el resultado del test moderno de aptitud del lenguaje (Carroll y Sapon, 1959), es decir, la facilidad de una persona de aprender un nuevo idioma.

1.5 Enseñanza y didáctica de la programación

Como afirma Litwin (1997) “entendemos a la didáctica como la teoría acerca de las prácticas de la enseñanza significadas en los contextos socio-históricos en que se inscriben” (p.95), lo que implica reconocer la importancia del contexto socio-histórico de las prácticas educativas y reflexión sobre las mismas. El contenido es determinante para la organización de las actividades, las que deben estructurarse en función del áreas del conocimiento y la lógica interna de la disciplina (Antúnez et al., 1992), dando lugar a lo que en la tradición uruguaya se han denominado didácticas específicas (Pesce, 2008). Las actividades a realizar están asociadas a las estrategias áulicas, las cuales tienen relación con formas de pensar y aprender, al tiempo que son mediadores entre los resultados y las intenciones docentes (Fiore y Leymonié, 2007).

Paradigma de programación, lenguaje, modelado y estrategias no son otra cosa que facetas de la didáctica de la informática; la cual tiene a la computadora, el modelado y la programación como elementos estructurantes (Miños, 2017). En

efecto, las variables anteriores determinan las prácticas educativas, las cuales están condicionadas por el contexto sociocultural, aspectos propios de la didáctica en el sentido dado por Litwin (1996). La importancia de la programación y su incidencia en casi todas las áreas de la informática (Denning et al., 1989) es tal que solo se pueden comprender la concepción ingenieril de la segunda comprendiendo a su vez la primera, en particular en lo relacionado con su enseñanza. Por tanto desde la informática como disciplina científica y su didáctica, se impone el estudio de la enseñanza y aprendizaje de la programación.

No obstante lo anterior, la didáctica de la informática tiene una débil tradición, confundiéndose en ocasiones con la didáctica de la tecnología, con el estudio de aspectos exclusivamente epistémicos o incluso limitándose a la enseñanza. Estas dificultades dieron lugar a visiones contrapuestas sobre cuál era el objeto de estudio de la informática o si el mismo era o no parte de otro constructo, teniendo esto incidencia en la propia construcción del objeto. Por ejemplo, Denning et al. (1989) sostiene que decir que informática y programación son sinónimos es una idea engañosa. Por otro lado, mientras Dijkstra relaciona fuertemente la programación con la matemática y la lógica (Denning, 1989), van Emden afirma que:

> "People who know neither programming nor mathematics (i.e., almost everyone) take for granted that programming is like mathematics. Yet, it turns out that English majors are as likely to be as successful at programming as mathematics graduates are. In practice, the worlds of mathematics and programming are just about disjoint. Dijkstra argues that this should not be the case, that mathematics is a formal game of symbol manipulation and that programming should become one. I must confess that the role of symbol manipulation in mathematics has me thoroughly confused." (Denning, 1989, p. 1407).

Esta construcción histórica de la didáctica de la informática tiene un elemento importante con la diferenciación entre la construcción de algoritmos y lograr que estos puedan ser ejecutados por una computadora. En efecto, en 1988 Arsac afirma que la didáctica de la programación no es lo mismo que algoritmia, la cual trata del estudio de los algoritmos, sino que la didáctica se reduce a analizar cómo resolver problemas que a su vez deben ser ejecutados por una computadora. Posteriormente Kaasbøl (1998) analiza las aproximaciones didácticas a la enseñanza de la programación, describiendo tres grandes formas denominadas Escalera semiótica (sintaxis, semántica, pragmática), Taxonomía

de objetivos cognitivos (correr, leer, cambiar y crear un programa) y Resolución de problemas; centradas las dos primeras en la enseñanza y la tercera en el aprendizaje.

Como adelantamos previamente, desde la didáctica de la informática en enseñanza media tres dimensiones cobran una importancia relevante: la computadora (u otro ente que actúe como tal), la construcción de modelos y la programación (Miños, 2017). La computadora asegura que el estudiante pueda verificar si las soluciones que plantea son adecuadas para la realidad dada (Dowek, 2005). Sin computadora el docente debe recurrir a subterfugios de la forma de "corridas a mano" o "asumir que se es un equipo informático", pues la persona no puede comportarse como el equipo mencionado; ocultando los eventuales errores que existen en el programa. Obsérvese el hecho que informática no es lo mismo que computadora, pero sin embargo desde la didáctica de la disciplina es fundamental esta última. El modelado es otra de las dimensiones de la didáctica de la informática, pues aquello computable no es otra cosa que parte de la realidad, que debe ser representada para lograr soluciones eficientes. Por último reiteramos lo dicho en el sentido que la programación es transversal a los saberes y planes de estudio (Denning et al., 1989), por lo que el estudio de aprendizaje y enseñanza es también el estudio de la didáctica de la informática.

El criterio para determinar qué características tiene un programa para ser correcto está sujeto a las intenciones y formas de trabajo de enseñanza. La inicialización de una variable, por ejemplo, si bien puede considerarse un elemento conceptual importante, está condicionado por el lenguaje de trabajo. En efecto, si el lenguaje inicializa las variables y en un programa esa inicialización coincide con el valor necesario, podría no considerarse un error la ausencia de la inicialización. Más aún, el uso de un conjunto de variables (como byte, entero sin signo, u otro) también está condicionado por el elemento tecnológico y la disponibilidad de memoria, afectando así las actividades de enseñanza. Es así que a priori no podríamos estar demostrando la existencia de invariantes en relación a qué elementos considerar a la hora de evaluar la correctitud de un programa. No obstante lo anterior, podríamos acordar qué aspectos son necesarios considerar para evaluar la correctitud de un programa que resuelve un problema en particular; aspecto que hemos logrado consensuar en este trabajo en función de lo dicho por varios profesores . Dado que un programa es viable en el contexto que el autor le da, podemos decir que en general esto depende de si el mismo presenta las bases adecuadas para su funcionamiento, o lo que es lo mismo, si la cantidad y calidad de errores presentes es tal que no comprometa su lógica de resolución del problema.

Soloway (1986) analiza lo que serían formas que permitan establecer un "plan" de enseñanza de la programación; determinando posibles formas de trabajo, refinamiento y sintaxis. En línea con el trabajo previo, Ebrahimi (1994) estudia qué tipo de errores ocurren en la construcción de un programa que itera con una condición de finalización, concluyendo que la construcción de un plan adecuado está asociado a programas viables. La bibliografía consultada evidencia que el gran desafío en la enseñanza de la programación lo constituye la construcción de planes y programas funcionales, pues los programadores noveles conocen la sintaxis y semántica individual de las sentencias a usar, pero no son capaces de organizarlas en un producto que resuelva el problema (Winslow, 1996). El desconocimiento de la semántica de las sentencias, su significado y utilidad, es un factor que afecta negativamente la explicación e interpretación de un código por parte de los alumnos (Xie, Nelson y Ko, 2018).

En el caso de los programadores noveles, y tomando como base una población de 559 estudiantes que realizan cursos en C++, Lahtinen, Ala-Mutka y Järvinen (2005) concluyen que el aprendizaje de contenidos como variables, estructuras de control e iterativas, son los que muestran mayor dificultad de aprendizaje. Los investigadores afirman que el aprendizaje de conceptos propios de programación no resulta tan difícil como la combinación de los mismos que da lugar a programas mayores; concluyendo que si bien es importante la teorización sobre dichos conceptos, sería necesario incrementar las actividades prácticas. En línea con lo anterior, y como una primera aproximación a los contenidos a trabajar, sería adecuado considerar la construcción de actividades que involucren pocas sentencias nuevas y con una relativa simplicidad semántica. En tal sentido una opción de trabajo lo constituye la realización de actividades de tipo inductivas, con baja dificultad semántica y sintáctica para los estudiantes, las cuales en un estudio de caso han resultado atractivas para los estudiantes (Miños, 2016a).

Otra de las dimensiones a considerar la constituye los denominados modelos mentales (Van Dijk, 2005), esto es, representaciones internas que hacen las personas sobre conceptos o hechos, los cuales son persistentes en el tiempo a la vez que resistentes al cambio (Fiore y Leymonié, 2007). Estas representaciones deben ser consideradas, pues los programadores nóveles tienden a incluir una multitud de errores que dificultan la construcción de soluciones viables. En el caso de modelos mentales asociados al manejo de variables, se ha observado que el uso de diagramas estaría ayudando a mejores interpretaciones de las asignaciones, aunque dicho extremo no se da necesariamente en un corto espacio de tiempo (Ma, Ferguson, Roper y Wood, 2011). Sanders, Galpin y Götschi (2006) analizan los modelos mentales de la programación recursiva, combinando para ello el análisis de los programas

construidos por los estudiantes con la interpretación que hacen los mismos y representan de forma verbal.

La contextualización del problema a resolver es otra de las variables a tener en cuenta desde la didáctica de la informática. La construcción de una actividad puede ser un fin en sí mismo, permitiendo al alumno manipular estructuras de control, variables u otros conceptos fuertes de la programación. Al mismo tiempo podríamos afirmar que la contextualización del problema podría dar lugar a mejores resultados, pues los estudiantes no solo podrían aplicar los conocimientos sino que además darle un sentido real. En tal sentido el proyecto Cupi2 (Villalobos y Calderón, 2009) propone la estructuración de un curso de nivel terciario en función de ejemplos concretos que permitan contextualizar los aprendizajes, entre otros aspectos. Sin embargo, en un estudio que involucró a cinco instituciones educativas, Bouvier et al. (2016) concluyen que al menos en niveles de educación terciarios, la contextualización del problema a resolver no tiene incidencia en la correctitud de las soluciones propuestas a ejercicios, aunque no descartan los efectos positivos en la motivación y predisposición a estudiar. No obstante lo anterior, Bouvier et al. (2016) sostienen que los resultados no descartan los efectos que puede tener la motivación en las intenciones de aprendizaje de los estudiantes y la realización de las tareas, pudiendo ser una variable a estudiar.

La resolución de un problema, la abstracción y automatización parecen invariantes en aquello que podríamos definir como didáctica de la informática. Programar implica resolver problemas computacionales de forma eficiente, considerando por tanto los recursos en general; dimensiones análogas a las del pensamiento computacional en el sentido dado por Wing (2006). Aunque pensamiento computacional no es lo mismo que computación ni programación, existen varias formas de desarrollar el primero, siendo una de estas el aprendizaje de la programación (Resnik et al., 2009), la cual a su vez hace a la programación su mejor exponente (Compañ-Rosique et al., 2015). Aho (2012) sintetiza la relación entre computación y pensamiento computacional pues “computation is a process that is defined in terms of an underlying model of computation and computational thinking is the thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms” (2012, p. 832).

1.6 Scratch como lenguaje de programación

Uno de los posibles lenguajes de programación a usar en enseñanza media es Scratch, el cual cumple con las características de un buen primer lenguaje de: piso bajo, techo alto y paredes anchas, según Papert; dicho de otro modo, el lenguaje permite que los estudiantes lo aprendan fácilmente, a la vez que se logra construir programas cada vez más complejos, aplicables a distintos tipos de proyecto (Resnik et al. 2009). Scratch resulta intuitivo, con una sintaxis simple, interfaz amigable, con una reducida cantidad de constructores y que puede usarse en otras áreas de la informática (Maloney, Resnick, Rusk, Silverman y Eastmond, 2010); lo que lo convierte en un lenguaje de fácil aprendizaje, particularmente en enseñanza primaria y secundaria. En tal sentido, la percepción que tienen los estudiantes de la facilidad de uso, lo divertido de trabajar con el lenguaje o qué tan interesante resulta su uso (Kaučič y Asič, 2011), lo convierten en un muy buen primer lenguaje de programación, permitiendo a su vez reducir la carga cognitiva asociada a la resolución de problemas de bajo nivel, desde lo programacional, haciendo que el estudiante se centre en los aspectos más importantes a resolver (Flannery et al., 2013).

Aunque el lenguaje ha facilitado los aprendizajes de programación, se ha observado que los programas de asignaturas, del equivalente a ciclo básico, no hacen hincapié en los conceptos fuertes de programación, presentando los alumnos grandes falencias en conceptos fuertes como el manejo de variables (Grover y Basu, 2017). En estudiantes de ciclo básico, y a nivel de aula, los mismos no siempre logran aprender conceptos básicos de programación como iteraciones o el diseño de algoritmos (Meerbaum-Salant, Armoni y Ben-Ari, 2011), motivado ello parcialmente por la tendencia a la concretización que evidencian los estudiantes, quienes adaptan y simplifican el nuevo conocimiento para darle cabida en su estructura cognitiva (Meerbaum-Salant, Armoni y Ben-Ari, 2013).

Incluso considerando las debilidades del lenguaje o de las propuestas de trabajo de aula, Scratch permite el desarrollo de habilidades computacionales y comunicacionales (Scaffidi y Chambers, 2012), pudiendo desarrollarse las mismas mediante actividades lúdicas (Vázquez-Cano y Delgado, 2015), las cuales son una forma de fomentar la motivación de los estudiantes en la actividad.

La interrelación entre Scratch y otras áreas del conocimiento no se limita a la programación o matemática, como podríamos pensar en un principio. En un estudio de 18 meses de duración que involucró 536 proyectos, se evidencia la

relación de la programación en Scratch con otras áreas del conocimiento como ser historia, matemática o lengua entre otros (Maloney et al., 2008). El mismo estudio concluye que los alumnos asocian fuertemente la programación con Scratch con actividades que fomentan la creatividad, como el arte y el lenguaje.

La versatilidad del lenguaje es tal que el mismo es usado en distintos niveles del sistema educativo, en ocasiones como un fin en tanto que otras veces como un medio. Existen experiencias asociadas a la inclusión de Scratch como lenguaje de programación en cursos introductorios de programación en nivel universitario, las cuales resultan motivantes para los estudiantes (Muñoz et al., 2015) y permiten el aprendizaje rápido de conceptos fuertes de la programación (Wolz, Leitner, Malan y Maloney, 2009). Cuando Scratch es usado en niveles secundarios, con anterioridad a lenguajes como C# o Java, se observa que los estudiantes tienden a aprender más fácilmente algunos conceptos propios de la programación (Armoni, Meerbaum-Salant y Ben-Ari, 2015). Sin embargo debe tenerse en cuenta que la mera inclusión del lenguaje no es condición suficiente para comprender todos los conceptos fuertes de la programación, afirman los autores del trabajo.

1.7 Tema y problema

El aprendizaje de la programación es una actividad no trivial, asociada a varias actividades cognitivas, que además requiere hacer uso de estrategias flexibles de aprendizaje (Rogalski y Samurçay, 1990), donde los estudiantes no solamente resuelven problemas, sino que buscan que una computadora lo haga (Arsac, 1988). Rogalski y Samurçay (1990) afirman que no hay actividades cotidianas que preparen para al estudiante para aprender conceptos básicos de programación (como el manejo de variables o la recursividad), mientras que el funcionamiento de la máquina física donde se ejecuta el programa no siempre es claro. Es así que resulta particularmente difícil para un estudiante comprender cómo se debe programar, es decir, que debe entender y modelar un problema para luego codificar una solución que es resuelta por una computadora en un lenguaje dado.

Frente a lo anterior destacan algunas dimensiones a ser tenidas en cuenta al momento de planificar actividades de enseñanza y aprendizaje de la programación. La elección del lenguaje resulta fundamental (Fernández et al., 2002) pues el mismo se asocia a un paradigma de programación y por tanto a una forma de comprender y modelar el problema y la solución. La simplicidad

sintáctica de Scratch (Resnik et al. 2009) lo hace adecuado como primer lenguaje de programación (Maloney et al., 2010), sin que esto implique una simplificación tal que impida el aprendizaje de la programación. En efecto, el lenguaje ha permitido el aprendizaje de conceptos fuertes de la programación (Wolz et al., 2009), pudiendo ser usado como un antecesor exitoso de lenguajes de uso industrial (Armoni et al., 2015).

Considerando la enseñanza de la programación destacan algunos aspectos de relevancia a tener en cuenta, que si bien hemos desarrollado previamente, entendemos importante reiterar. Aunque el estudio del lenguaje de programación a usar es fundamental, esto no debe dejar de lado las estrategias áulicas (Fiore y Leymonié, 2007), así, las formas de representar y modelar los problemas. La identificación de los grandes elementos constitutivos del problema (Soloway, 1986; Ebrahimi 1994) es fundamental a la hora de construir programas y de enseñar a cómo hacerlo. De este modo el estudiante o programador deberá coordinar varios planes en uno único, siendo complejas las relaciones semánticas entre los distintos planes que se construyen, algo que resulta difícil de comprender para los nóveles programadores (Winslow, 1996). No obstante lo anterior, identificar planes viables no es suficiente para lograr aprendizajes en los estudiantes, siendo un aspecto a tener en cuenta qué actividades resultan motivantes para los estudiantes, destacando en ciclo básico aquellas de tipo lúdico (Vázquez-Cano y Delgado, 2015; Willing, Astudillo y Bast, 2010) y las propuestas asociadas al trabajo con el código con distintos grados de completitud (Kordaki, 2012).

Desde lo metodológico, Fitzgerald, Simon y Thomas (2005) analizan cómo los estudiantes de un grupo de cursos introductorios de programación a nivel terciario, explican verbalmente un código presentado, destacándose las siguientes categorías: reconocimiento de la sintaxis de datos por sobre la semántica de los mismos; reconocimiento de la semántica incluso si se desconocen aspectos del programa; uso y reconocimiento de patrones (sintáctico y semántico); recorridas de código. Renumol, Janakiram y Jayaprakash (2010) recurren también al análisis verbal de las explicaciones que realizan estudiantes mientras programan, infiriendo una serie de procesos (denominados procesos cognitivos en el artículo) que según los autores son puestos en juego, como ser confusión, iteración e hipótesis entre otros.

El dibujo de bocetos es una estrategia usada en los cursos de programación como forma de comprender el funcionamiento de un programa, rastreando la asignación de valores en las variables. Cunningham, Blanchard, Ericson y Guzdial (2017) afirman que “leaving a problem blank was associated with the

lowest average score of any sketch type… Choosing not to sketch is not a successful strategy for solving code reading problems.” (p. 167), sin que esto signifique que cualquier tipo de boceto implique los mismos resultados.

Xie et al. (2019) analizan varios aspectos en relación a la enseñanza de la programación, destacando en particular el hecho que el estudiante debe ser capaz de leer el código en un lenguaje en particular (semántica de lectura), además de poder escribir las sentencias en el lenguaje dado (semántica de escritura). La relación entre ambas semánticas es fuerte, pues un estudiante que posee un amplio conocimiento de semántica de lectura pero no de escritura estaría construyendo programas cercanos al pseudocódigo, no funcionales. Inversamente, un estudiante con una buena semántica de escritura y baja de lectura, escribiría código sintácticamente correcto pero que no estaría resolviendo el problema.

Brennan y Resnick (2012) recurren al análisis de programas y entrevistas a estudiantes y profesores con el fin de conocer su opinión sobre la programación en Scratch, en particular aspectos de diseño y de resolución del problema. Los autores concluyen que los estudiantes prefieren prácticas iterativas e incrementales, prueba y depuración, reutilización y remezcla, abstracción y modularización.

En los trabajos analizados hemos observado que el lenguaje es usado como un instrumento que permite conocer la interpretación que hacen los estudiantes del hecho, en ocasiones usando el lenguaje verbal y en otras el lenguaje escrito (diagramas, bosquejos en incluso lenguaje natural). Si bien hemos relevado documentos en los cuales se pide a los alumnos que expliquen cómo se resolvería un programa, hemos visto que el tamaño relativo de los mismos no permite identificar claramente el modelo según nuestra interpretación del mismo. Es por lo dicho que entendemos que no hemos encontrado artículos que relacionen el modelado del problema en el sentido dado en este trabajo (el cual es previo a la programación efectiva del programa) con el uso del lenguaje natural. En tal sentido Lye y Koh (2014) afirman que sería interesante analizar cómo los estudiantes verbalizan el proceso cognitivo, al tiempo que sostienen que la mayoría de las investigaciones analizadas se centran en el aprendizaje de conceptos asociados al pensamiento computacional. Esta concepción de modelado no es el conjunto de técnicas propias de la representación de programas, sino que tienen un enfoque de más alto nivel, orientadas al análisis de la realidad que no necesariamente tienen relación con las técnicas de programación.

Se evidencia una preocupación por identificar y analizar aquellas formas relacionadas con la enseñanza y aprendizaje de la programación, en todos los niveles educativos, maximizado esto por la relación con el pensamiento computacional (Compañ-Rosique et al., 2015; Resnik et al., 2009; Aho, 2012). En efecto, programar implica abstraer, construir modelos y pensarlos desde la eficiencia, aspectos propios del pensamiento computacional. Resulta importante reconocer aquellos aspectos que hacen al proceso de construcción de algoritmos en general y de la programación en particular, lo cual se da en dos niveles distintos y complementarios: a nivel micro, esto es, el estudio de la construcción de los programas; y a nivel macro la forma como es modelado o representado el problema en una etapa previa a la construcción efectiva del programa. Mientras la primera dimensión tendrá como objeto de estudio los programas, la segunda se centrará en las representaciones que hacen los estudiantes del objeto a construir (el programa) para luego analizar la correspondencia entre el producto y su representación. Es así que el análisis debe ser realizado considerando la importancia e implicancias que tiene el modelado para las dimensiones de la didáctica de la informática (Miños, 2017), aspecto propio de las didácticas específicas (Pesce, 2008).

Por último recordemos que cuando hablamos de informática como asignatura en Ciclo Básico, la misma se dicta tanto en el ámbito de la Dirección General de Enseñanza Secundaria (DGES, 2016) como en la Dirección General de Educación Técnico Profesional (DGETP, s.f.) y en esta última Dirección con un perfil levemente profesionalizante. Si bien los cursos que se dictan en ambos ciclos son equivalentes, en el caso de la DGETP la asignatura está inserta en cursos que buscan formar a los estudiantes de rápida inserción laboral. Por tanto, la informática en la órbita de la DGETP se relaciona con otras asignaturas orientadas al mercado laboral, en tanto que en DGES con asignaturas que buscan una alfabetización científica o la denominada cultura general.

De este modo queda definido el problema de investigación:

> El proceso de construcción de programar está íntimamente relacionado con la construcción del modelo del problema, aspecto que está condicionado a su vez por los saberes disciplinares del estudiante. En contextos educativos donde la enseñanza de la programación no está orientada al mercado laboral y por tanto no son un fin en sí mismo, las destrezas que posee un alumno para modelar el problema se ven limitadas por los saberes previos que posee y su estructura cognitiva. Siendo necesario modelar el problema como paso previo a la programación, entendemos que es necesario

profundizar el estudio de cómo los estudiantes modelan un programa; es decir, cómo es el proceso de representación del mismo en etapas anteriores a la programación propiamente dicha.

1.8 Preguntas de investigación

Como mencionamos previamente existen varios trabajos que analizan el discurso de los estudiantes luego de haber construido el programa, justificando sus elecciones y explicándolas. También mencionamos trabajos en los cuales los estudiantes verbalizan su solución mientras la misma es construida. Sin embargo, para estudiantes de un nivel secundario básico (dos primeros años de enseñanza post primaria, con una población de adolescentes) reiteramos lo dicho en el sentido que no hemos encontrado trabajos en los cuales se analiza el modelado de soluciones programacionales con anterioridad a la programación; aspectos estos mencionados por Lye y Koh (2014).

Por lo expuesto entendemos que el presente trabajo adquiere un enfoque exploratorio, donde no es posible establecer una hipótesis a validar por la propia ausencia de trabajos previos que determinen una línea de trabajo sólida. No obstante lo anterior, sí es posible definir algunas preguntas a ser respondidas y que guiarán nuestro trabajo, a saber:

1. ¿Cómo expresan los estudiantes el modelo de un problema a resolver?
2. ¿Qué conceptos se asocian al modelado de un problema?
3. ¿Qué relación hay entre los elementos constitutivos del problema y la forma de expresarlo por parte de los alumnos?
4. ¿Qué relación existe entre las soluciones construidas y la forma de ser expresado en lenguaje natural el modelado del problema?
5. ¿Es posible evaluar automáticamente la capacidad de modelar un problema de programación a resolver a partir del discurso de los estudiantes?

Capítulo 2

Diseño metodológico

2.1 Población y contexto

Nuestro trabajo de campo tuvo como población objetivo estudiantes de Ciclo básico que se encuentran cursando los dos primeros años de la enseñanza post primaria, bajo la órbita de la Dirección General de Enseñanza Secundaria. Tanto primer como segundo año de Ciclo básico tienen una carga horaria de 39 horas pedagógicas semanales, con un total de 13 asignaturas para cada uno de los años, donde una de las mismas es Taller de Informática (DGES, 2016).

Si bien se realizaron tres grandes actividades, no todos los estudiantes las completaron, por lo que no fueron considerados en el análisis de datos posterior. La población estudiada, y por tanto quienes completaron todas las tareas, fue de 94 alumnos, siendo 59 varones y 35 mujeres, entre 13 y 15 años, cursando en el Liceo número 38 de Montevideo, en el barrio de La Teja. No se tiene registro de la existencia de dificultades especiales de aprendizaje, en particular las relacionadas a la escritura o comprensión lectora.

Las actividades se realizaron durante el año lectivo 2020, por lo que no debemos dejar de considerar el contexto de pandemia por el Covid - 19. Durante el año 2020 en la enseñanza secundaria del sistema educativo nacional, las actividades de enseñanza se realizaron en una modalidad mixta. Durante la primera parte del año todos los cursos siguieron una modalidad virtual, donde cada uno de los docentes generaba las instancias y actividades que entendía necesarias para

lograr los aprendizajes en los estudiantes. Si bien existieron pautas de las inspecciones técnicas, las mismas eran orientaciones generales de trabajo y no de aplicación obligatoria. De este modo, mientras algunos docentes optaron por la realización frecuente de videoconferencias, otros en cambio fueron a una modalidad exclusivamente virtual, orientada al trabajo en plataforma y objetos de aprendizaje. Naturalmente existieron docentes que combinaron ambas propuestas, siendo este el caso del Prof. Herrera, quien nos permitió realizar las actividades de campo en sus grupos. Durante el correr del año se habilitó la presencialidad parcial en Ciclo básico, con grupos reducidos, alternando la concurrencia a clase entre los estudiantes (generalmente se concurrió a clase semana por medio). De este modo no fue posible trabajar con la totalidad del grupo, por lo que cada una de las tres actividades macro se realizó en dos días; una jornada para la mitad del grupo y la siguiente para la otra mitad.

2.2 Problema y datos relevados

2.2.1 Problema y variables de la evaluación

La recolección de datos para nuestro trabajo se dividió en tres grandes actividades, las cuales se asocian al análisis de datos desde dos grandes perspectivas. Comenzamos con una actividad en la cual se solicitó la explicación de una serie de conceptos teóricos; luego se pidió explicación sobre cómo se haría un programa dado; por último solicitamos la realización del programa anterior para posteriormente corregirlo. Las variables obtenidas de las dos primeras actividades se sujetaron a una análisis semántico y estadístico, usando entre otras herramientas de machine learning, en tanto que el programa fue evaluado siguiendo criterios estrictamente computacionales.

En la primera actividad planteada realizamos una serie de preguntas teóricas a los alumnos, cada una asociada a una consigna de trabajo: uso de variables, de estructuras de repetición y de la sentencia (tabla 1). Evaluadas las respuestas, a las variables les fue asignado tres posibles valores: 2, 1 o 0 en función de su correctitud. Asignamos 2 a las respuestas que desarrollaron correctamente el fenómeno, hecho o concepto, así como sus características. El valor 1 fue dado a las respuestas que presentaban falencias relativamente importantes al explicar el concepto o hecho o bien a los alumnos que responden en función de un ejemplo, sin profundizar o generalizarlo. 0 fue asignado a las respuestas que evidencian errores muy importantes, que presentan un ejemplo que no tiene relación con la consigna o que no responden a la consigna. Los puntajes obtenidos fueron

asignadas a diferentes variables (tabla 1). Estas variables se identifican con los nombres R_VARIABLES, R_REPETICIÓN y R_CONDICIÓN. Una tercera variable es la suma de los valores obtenidos en las preguntas R_VARIABLES, R_REPETICIÓN y R_CONDICIÓN, que toma el nombre SUMA_R. De este modo, si R_VARIABLES, R_REPETICIÓN y R_CONDICIÓN toman los valores 2, 1 y 2 respectivamente, SUMA_R tomará el valor 5.

VARIABLES CONSIDERADAS	
CONSIGNA	VARIABLE DEFINIDA
Cuando vas a hacer un programa, ¿cómo te das cuenta que debes usar una variable?	R_VARIABLES
Cuando vas a hacer un programa, ¿cómo te das cuenta que debes usar una estructura de repetición?	R_REPETICIÓN
Cuando vas a hacer un programa, ¿cómo te das cuenta que debes usar una instrucción Si?	R_CONDICIÓN
Cuéntale a alguien cómo resolverías el programa siguiente:	T_OBJETOS, T_MOVIMIENTOS, T_VARIABLES y T_CONDICIONES
PROBLEMA	
Uso de: inicialización de posición y variables (2), movimiento del vehículo (3), giro fluido del vehículo (0,5), movimiento inercial del vehículo (0,5), salida del circuito y finalización del juego (1,5), finalización del juego por llegada a la meta (1,5), uso de variable y suma de puntos (2) y destrucción del vehículo (1).	Se califica el trabajo de 1 a 12.

SUMA_R es la suma de los valores obtenidos con R_VARIABLES, R_REPETICIÓN y R_CONDICIÓN.	SUMA_R
SUMA_T es la suma de los valores obtenidos con T_OBJETOS, T_MOVIMIENTOS, T_VARIABLES y T_CONDICIONES.	SUMA_T

Tabla 1: variables consideradas

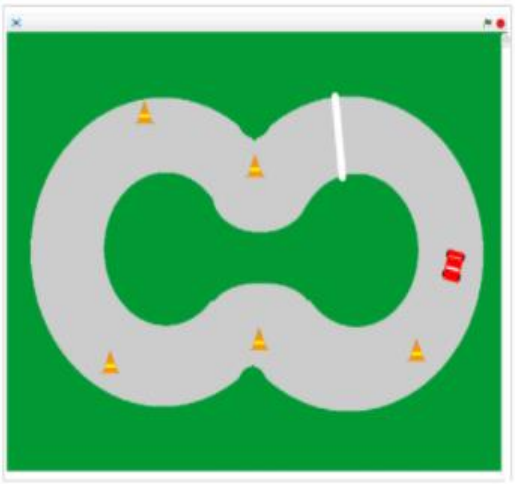
Problema:

"Se quiere programar un juego en el cual se controla el movimiento de un auto para que se mueva por un circuito.

Al salir del mismo el auto se destruye y finaliza el juego.

El auto avanza solo de a 2 pasos.

Los puntos se logran al pisar con el auto una serie de marcas en el circuito, obteniendo 10 puntos por cada marca.



Al llegar al fin (marcado con una línea) se indica que ganó y finaliza el juego mostrando los puntos. "

Figura 1: consigna e imagen entregada a los estudiantes

Luego presentamos a los estudiantes la imagen de figura 1 y les pedimos que expliquen cómo harían para resolver el problema allí planteado, lo que implicaría incluir aspectos propios del modelado del problema. Dado que no se dispone de información suficiente como para inferir cómo es el modelado de parte de los estudiantes de Ciclo básico, es que se optó por preguntas de respuesta abierta, pues las mismas son útiles cuando no se dispone de información suficiente del objeto de estudio (Hernández, Fernández y Baptista, 1991). Frente a la explicación escrita de los alumnos, procedimos a analizar los resultados y agruparlos en función de las siguientes variables:

1. T_OBJETOS. Variable relacionada con la explicación de la relación existente entre los objetos: existencia de un vehículo, necesidad de programarlo y su interacción con otros elementos del sistema. Dado que los conos existentes en el juego podrían no ser considerados como objetos, es que la evaluación de la ocurrencia de los mismos fue considerada en función del contexto.

2. T_MOVIMIENTOS. Esta variable evalúa la explicación de cómo sería la programación de los movimientos: existencia de relaciones entre el movimiento del vehículo, las acciones del usuario y su programación (uso de teclado o ratón).

3. T_VARIABLES. Variable que evalúa el uso de variables: identificación de la necesidad de la misma y las operaciones asociadas (suma de valores o cuenta en su defecto, aceptando esto último como un abuso de lenguaje).

4. T_CONDICIONES. La existencia de condiciones fue considerada en esta variable, la cual se asocia a salir del circuito, tocar un color para finalizar el juego, sumar puntos o condiciones de victoria.

5. SUMA_T. Esta variable es análoga a SUMA_R, siendo la suma aritmética de las 4 variables anteriores. Asumimos que valores altos de SUMA_T se relacionan con explicaciones globalmente adecuadas.

Cabe mencionar el criterio que llevó a considerar las variables T_OBJETOS, T_MOVIEMIENTOS, T_VARIABLES y T_CONDICIONES como determinantes en este problema. La propia solución implica reconocer y programar la existencia de objetos que deben interactuar entre si y con el usuario, mientras se evalúan una serie de condiciones que llevan al fin del juego y modificar una serie de datos. Es por lo anterior que surge la necesidad de operar con las cuatro variables anteriores, las cuales se relacionan con una buena programación. No obstante lo anterior, es cierto que existen otros elementos que pueden ser considerados como fundamentales en la programación, como ser la inicialización de variables y coordenadas de inicio o la fluidez del movimiento. Sin embargo, las dimensiones anteriores no son excluyentes para un correcto funcionamiento del programa (aunque su omisión o incorrecta programación llevarían a programas no eficientes), en tanto que aquella asociadas a las variables son determinantes. Dicho de otro modo, las cuatro dimensiones que se asocian a cuatro variables, son tales que es condición necesaria su correcta identificación para la construcción de un programa viable. Cabe mencionar que realizadas consultas con otros docentes (ver anexos), los mismos expresaron una solución

al problema en las cuales se evidencia que la misma también identifica las mismas dimensiones que consideramos en este trabajo.

Por último realizamos una actividad de programación con los estudiantes, individual y obligatoria. El problema a resolver fue el mismo que presentado en la consigna anterior. El ejercicio fue evaluado de 1 a 12 según los siguientes indicadores: inicialización de variables y posición (2 puntos), movimiento del vehículo (3 puntos), movimiento inercial (0,5 puntos), giro fluido (0,5 puntos), salida del circuito y fin del juego (1,5 puntos), condición de finalización normal (1,5 puntos), manejo de puntos (2 puntos), destrucción de vehículo (1 punto). Calificar un producto, en este caso de software construido por un estudiante, es un aspecto que sirve como indicador de logro, traduciendo la correctitud de dicho producto en función de una serie de parámetros. Es así que calificar se convierte en una reducción metodológica, que permite cuantificar el propio producto. La calificación del programa cumple la función sumativa de la evaluación (Fiore y Leymonié, 2007), que siendo un indicador de la consecución de los objetivos, en este caso la apropiación de los saberes disciplinares necesarios para resolver un programa.

Podemos observar que en nuestro caso y en general en toda actividad evaluativa de la programación, se incluyen aspectos que no forman parte del modelo, pero que sin embargo dan cuenta de una construcción prolija o adecuada de un programa. De las categorías consideradas en la evaluación del programa, al menos la inicialización del programa, movimiento inercial y giro fluído del vehículo representan 3 puntos de la calificación y a nivel de diseño no tendrían relación con las respuestas teóricas. Dicho de otro modo, un estudiante puede obtener calificaciones altas o bajas independientemente de haber modelado verbalmente estas categorías. Sin embargo debemos tener en cuenta que estas dimensiones podrían ser un indicador indirecto de un correcto modelado, sin que sean expresados en el mismo. En efecto, estos aspectos podrían estar tan internalizados por el estudiante que formando parte de la solución efectiva no sería verbalizadas. Por tanto, estos aspectos se relacionan con una correcta programación, pero no estarían incluidas en la verbalización del modelado. Debemos tener en cuenta que lo previamente expresado está en línea con las representaciones hechas por los docentes a quienes solicitamos opinión y que se encuentran en los anexos.

En el caso de la salida del circuito y finalización del juego, las mismas se asocian a dos conceptos y procedimientos importantes: la identificación de una condición y las acciones asociadas a la misma (detención del programa, reinicio o muestra de resultado). Por tanto, mientras un alumno podría explicar razonablemente

bien el uso de condiciones (de salida del circuito y finalización del programa), esto no significa que pueda programar otras acciones, como detener el programa (con el eventual uso de mensajes o formas coordinación entre varios objetos), mostrar los puntos, manipular colisiones de objetos o resolverlo mediante tocar un color. Es así que de los 3 puntos en total de estas categorías, al menos 1,5 se asocian a la programación posterior y la identificación de la condición.

En el caso de variables debe considerarse que la misma debe ser inicializada, no debe existir loop (por la superposición del objeto o color del cono) y actualización efectiva en su valor. La explicación de cómo usar variable, no está necesariamente en relación con su correcta manipulación, siendo un error recurrente la sobre actualización de la variable, aspecto que reduce en 1 punto la calificación máxima.

En el caso de la destrucción de un vehículo el mismo se asocia a modificar un disfraz y existir una condición. Si bien existe una relación con T_CONDICIONES, se verificó un grado de concretización significativamente alto, que no se mapea directamente con el concepto, centrándose los estudiantes en el uso de ejemplos más que en la explicación de qué es una condición.

Por último decir que la respuesta T_OBJETOS no se relaciona a un tipo de solución en particular, pues existe la posibilidad de usar varios tipos de objetos (conos y automóvil) o bien un solo objeto (auto) y manejar los objetos en función del color anaranjado.

De este modo es que podemos afirmar que si bien existe relación entre las respuestas sobre cómo se resolvería el problema, dado esto por las preguntas T y la calificación obtenida, no es posible afirmar que la explicación de cómo resolver el problema se asocie automáticamente a la calificación del programa.

2.2.2 Ejemplos de respuestas y tabulación de datos

A modo de ejemplos presentamos algunas respuestas obtenidas de parte de los alumnos y cómo ha sido su valoración según cada variable.

Comenzamos ejemplificando respuestas a las preguntas de conceptos teóricos, que fueron identificadas como R_VARIABLES, R_REPETICIÓN y R_CONDICIÓN. Cada oración corresponde a una respuesta distinta, manteniendo en todos los casos la redacción original de los estudiantes.

Para el caso de las variables los estudiantes deberían explicar que una variable se usa para almacenar un valor o que el mismo puede cambiar; aspectos como el tipo de datos, la volatilidad o el nombre no son relevantes en este caso, entre otras cosas por cómo maneja Scratch las variables.

Este primer grupo de estudiantes fue evaluado en R_VARIABLES con 2 puntos. Estos alumnos mostraron respuestas correctas, las cuales explican razonablemente bien cómo es posible darse cuenta que se debe usar una variable.

> “tengo que usar una variable para guardar un valor que se puede modificar”. (sic)

> “cuando queres poner un numero que quieras que se cambie con una accion o etc, una variable es un numero que se puede cambiar con una accion o etc” (sic).

> “la variable se neceita para por ejemplo a un videojuego le quereres poner puntuacion de como vas llendo por ejemplo si tengo que crear un videojuego de el gato cabezeando la pelota y quiero que cada ves q toque la pelota en la cabeza me sume un punto” (sic).

Un segundo grupo de estudiantes responden de forma general y se asignó 1 a R_VARIABLES. Como podemos ver en los ejemplos, estas respuestas hacen referencia a ejemplos y casos concretos de usos, como en el caso de las vidas. Incluso en el tercer estudiante, que menciona aspectos como la modificación del valor, obsérvese que la redacción suele ser imprecisa y poco clara en relación a la necesidad o conveniencia de usar una variable.

> “cuando x ejemplo necesitas un contador de vidas o algo por el estilo podes crear una variable” (sic).

> “se debe de usar una variable para cuando se necesite sumar o contar puntos” (sic).

> “Porque si se programa para un juego se usa para un valor numerico o una cantidad de monedas,vidas etc porque puede sr modificado” (sic).

Las respuestas evaluadas con 0 para la variable R_VARIABLES no responden a la consigna, o lo hacen de forma errónea.

> “la podes usar para ayudar a la ubicación de parecer por ejemplo en un juego de fútbol a donde tengas que cabezar para que aparezca en distintos lugares la pelota” (sic).

> “porque una variable nunca puede faltar ej: en un juego nunca puede faltar la variable” (sic).

> “lo hacemos si queremos que algo cambie mutuamente” (sic).

El concepto de iteración está asociado a repetir un conjunto de instrucciones, una cantidad finita o no de veces. Se podría agregar la existencia de condiciones que determinan la finalización de las iteraciones.

Las respuestas a R_REPETICIÓN evaluadas con 2 muestran la necesidad de repetir un bloque, acción o similar, una cantidad dada de veces; reconociendo también indirectamente la existencia de un patrón de repetición..

> “me doy cuenta de que debo usar una estructura de repetición cuando necesite que una acción se repita determinada cantidad de veces o infinitas veces” (sic).

> “Porque me permite hacer o cumplir un grupo de instrucciones mientras se cumpla la condición “por siempre si” en Scratch" o hasta que la instrucción deje de cumplirse “repetir hasta que” en Scratch"” (sic).

> “te das cuenta cuando el objeto debe repetir la acción mas de una vez” (sic).

La variable R_REPETICIÓN fue puntuada con 1 cuando la explicación fue poco clara o se centraba en ejemplos. Este grupo de respuestas se asociaron principalmente a casos particulares de ejecución, con un grado de concretización muy alto y una baja descripción del concepto y sus características.

> “por ejemplo:para repetir 4 veces que un auto se mueva 20 pasos y que gire 90°” (sic).

> “el repetir algo se usa cundo queres que un movimiento se repita por algo en especifico por ej vs queres que el gato al tocar el borde vulva a su lugar entonces pones los otros bloques y el repetir” (sic),

> “repetir hasta que se cumpla la condicion por ejemplo repetir hasta que el gato llegue al borde de la pantalla” (sic).

El valor 0 en R_REPETICIÓN se asoció a respuestas que no responden a la consigna o que presentan alguna idea asociada al concepto, pero con un alto grado de vaguedad.

> “cuando el problema dice que el personaje haga cierta hacion pero que en algún punto deje de hacerlo y haga otra cosa” (sic).

> “porque capas que fallas y tienes que hacerlo mas de una vez y le metes repeticiones” (sic).

> “por que necesitas que se repita la condición hasta un cierto punto o veces” (sic).

El tercer concepto fuerte era el de una condición, la cual se relaciona con optar entre dos posibilidades para luego realizar una operación. Términos como Verdadero o Falso fueron usados por los estudiantes para responder.

R_CONDICIÓN tomó el valor 2 en casos como los enunciados siguientes. Obsérvese que es común el uso del término Condición para definir qué es una condición.

> “cuando quieras que una acción se ejecute solo si pasa tal otra, a eso se le llama condición” (sic).

> “Cuando quiero que pase algo solo si se da una situación determinada” .

> “cuando hay una condición que cumplir por ejemplo si pasa tal cosa se realiza una cosa” (sic).

El segundo grupo de respuestas se asocia a valores 1 en la variable R_CONDICIÓN, que como se puede observar presenta una grado de ejemplificación muy alto, siendo la respuestas casi exclusivamente dicho ejemplo.

> “ejemplo tenes tocando borde si la pelota toca borde q vuelva a su punto de partida” (sic).

> “me doy cuenta que debo utilizar las instruccion si, cuando se que algo puede pasar, por ejemplo si estoy programando un juego con

> vidas y la unica forma de perderlas es chocando con el borde utilizo el "si toca el borde" pierde una vida y ahi lo utilizo” (sic).
>
> “te das cuenta cuando tengas que hacer una acción tocando algo por ejemplo cuando toques el borde rebotar o tele trasportar” (sic).

Para terminar mostramos ejemplos de respuestas a las que les fue asignado un valor 0, pues la misma no solo no es clara sino que además ni siquiera presenta un ejemplo claro.

> “Cuando tenes que darle una orden al personaje” (sic).
>
> “+ te das cuenta por la condición si del programa o por que lo necesitas para el programa” (sic).
>
> “cuando queres hacer la accion hasta q pase la condicion” (sic).

A continuación pasamos a ejemplificar los resultados obtenidos en las variables T_OBJETOS, T_MOVIMIENTOS, T_VARIABLES y T_CONDICIONES. En este caso presentamos la respuesta completa que dio el estudiante, para luego explicitar los valores que fueron asignados a cada una de las variables. Al igual que para las variables anteriores, se mantuvo la respuesta original del estudiante.

El primer estudiante describe convenientemente la existencia del vehículo y el cono, mencionando también una posible solución en función de un objeto meta. El estudiante también hace referencia a la inicialización de en función de las coordenadas, aspecto no evaluado, pero que denota un conocimiento de este concepto fuerte. Al mismo tiempo se establece correctamente las relaciones entre eventos y condiciones; además del manejo de las variables.

> “Primero le doy movimiento al auto con las flechas y le doy un lugar fijo para aparecer cuando inicia el juego. Despues creo una variable para los puntos de los conos colocandola en 0 al inicio y sumando de a 1 punto si se tocan los objetos de los conos. Luego indico que si se toca el objeto de la meta este mande un mensaje al escenario para cambiarlo a uno que diga algo como "ganaste!", y tambien envio el mensaje al objeto del auto para esperar un segundo (para que el escenario tenga tiempo de cambiar) y detengo el juego. Por ultimo pondria que si se toca el color verde de fuera de la pista el auto cambie a un disfraz destrozado y se detenga su programacion.” (sic),
>
> T_OBJETOS: 2

T_MOVIMIENTOS: 2

T_CONDICIONES: 2

T_VARIABLES: 2

Este segundo estudiante, al igual que el anterior, ha hecho referencia a la inicialización de las coordenadas, pero ha omitido la programación de los movimientos. Seguramente el estudiante ha considerado que explicar cómo se programa el movimiento de un vehículo (que en el mejor de los casos se puede reducir a 3 sentencias) es una actividad trivial para ser considerada.

> "bueno primero que nada creo o descargo el escenario con conos por distintas partes de la pista,creo una variable llamada: puntos,y abajo de la bandera verde pongo:fijar puntos a 0 ,y ir a x:0 y:0.Luego usaría una estructura de si entonces con la condición de tocando borde que haga referencia al circulo(pista).Adentro del si entonces pongo la condición en esa estructura pongo: Cambiar disfraz del auto y lo haría como si fuera una explosión. hasta ahí una parte del programa.Luego usaría otra estructura del si entonces,pero con la condición de tocando:color(naranja) que haga referencia a los conos.bueno adentro de esa estructura pondría: cambiar puntos:+1. Y por ultimo usaría la misma estructura del si entonces con la condición de tocando color pero esta vez (blanco) que haga referencia al final de la meta y que adentro de esa estructura este la orden de terminar programa" (sic).

T_OBJETOS: 2

T_MOVIMIENTOS: 0

T_CONDICIONES: 2

T_VARIABLES: 2

Un tercer estudiante ha obtenido calificación 10 en la construcción del programa, aunque presenta un modelado poco claro. La identificación de los objetos es clara (conos y automóvil), así como la programación de los movimientos. Sin embargo, al momento de expresar las condiciones el estudiante no las plantea de forma clara, sea porque las omite, sea porque las presenta como la propia letra del ejercicio. El estudiante omite el trabajo con variables..

> “habrian 2 tipos de objetos, 1 carro y 5 conos (al pasar por encima de los conos ganas puntos) el carro lo controlas con las flechas y tienes que dar un cierto numero de vueltas (si te sales de la pista el carro explota y pierdes) al terminar de dar las vueltas dadas si pasas por la meta ganas el juego.” (sic).
>
> T_OBJETOS: 2
>
> T_MOVIMIENTOS: 2
>
> T_CONDICIONES: 0
>
> T_VARIABLES: 0

El siguiente estudiante identifica los objetos presentes en el juego, pero de forma incompleta. Las otras categorías las puntuamos con 0 pues son una descripción del juego y de la consigna más que pautas que permitirían deducir o explicar cómo sería la programación.

> “Primero deberia colocar el escenario y los objetos necesarios. después empezar a armar el programa y cuando terminamos de armarlo lo verificamos usandolo . para hacerlo deberiamos colocar los conos y cuando llega al final termina el juego. El auto debe ser el objeto que esquive los conos, los conos hay que ponerlos por todo el escenario para que el auto cumpla con lo mencionado, pero antes de todo hay que elegir el escenario.” (sic).
>
> T_OBJETOS: 1
>
> T_MOVIMIENTOS: 0
>
> T_CONDICIONES: 0
>
> T_VARIABLES: 0

Por último presentamos una respuesta relativamente extensa en cantidad de palabras. Podemos observar que la descripción que hace el estudiante es el mismo enunciado, con la única excepción de la inclusión de las variables.

> “el auto rojo se mueve por el circuo, si el auto sale del circuito se destruye, si pasa por arriba de los conos suma un punto y al llegar al final de la pista el juego se termina. para eso se necesita hacer

primero que nada el escenario. 2do se necesitaria crear una variable para cuando el auto pase por los conos suma un punto y se necesita hacer que el auto al llegar a la final vuelva a comenzar el juego." (sic).

T_OBJETOS: 0

T_MOVIMIENTOS: 0

T_CONDICIONES: 0

T_VARIABLES: 1

Si bien nosotros tenemos criterios de correctitud del problema y qué respuestas esperábamos de los alumnos, aspectos que fueron usados en este trabajo, igualmente realizamos consultas a colegas en relación a cómo podría ser un buen modelo de solución. En tal sentido planteamos la misma consigna a otros profesores de la especialidad Informática, todos egresados de formación docente. Cabe destacar que a no ser en un solo caso, no existió la posibilidad de hacer aclaraciones de la letra como sí fue posible realizarlo a los estudiantes en clase. Aunque algunas respuestas de los profesores se centraron en el algoritmo, más que en la explicación de cómo resolver el problema en lenguaje natural, las mismas coinciden con nuestra visión de lo que es una buena respuesta para el problema planteado y que incluye las siguientes categorías:

1. Programación del movimiento del objeto.
2. Existencia de criterios de finalización (éxito o derrota), asociados estos a condiciones, como ser el que el vehículo toque un color dado.
3. Operación sobre una variable que suma en función del contacto del automóvil con el cono, la marca o el color anaranjado.

Algunos profesores mencionaron el cambio del disfraz del objeto en caso de tocar el vehículo la zona verde, relacionado esto con el hecho que se considera que el auto ha chocado. Otros docentes hicieron un paralelismo entre el concepto de sumar, incrementar y contar. Se presentan las respuestas de los profesores en los anexos.

Es un hecho que para un profesor de informática el programa solicitado resulta trivial. Transformar un enunciado que representa un problema simple al lenguaje

natural es un proceso que no siempre puede ser llevado a cabo; siendo la propia sencillez del problema su dificultad. En efecto, pensemos en un programador experto que debe explicar cómo sumar los valores numéricos de un vector usando lenguaje C: ¿explicará el programador que deben definirse e inicializarse cada una de las variables usadas en las iteraciones? o por el contrario, ¿se centrará en las iteraciones asumiendo que definir e inicializar variables es un dato conocido? Debemos tener en cuenta la dificultad que se le presenta al experto ponerse en lugar del novel programador y cómo este representa la información. Así, el experto asumirá como conocido y trivial la necesidad de inicializar una variable, no incluyéndose en la representación de la solución. De este modo sería de esperar un mayor grado de detalle en los alumnos que en el caso de los profesores..

2.3 Herramientas de análisis de datos

En el presente trabajo realizamos una serie de análisis de los datos obtenidos, algunos principalmente estadísticos y otros basados en el machine learning. La elección de estas herramientas se fundamentó en poder describir convenientemente las relaciones entre elementos a modelar y aspectos teóricos a considerar. A continuación explicamos las herramientas que hemos usado para el procesamiento de datos, justificando a su vez la elección de las mismas y su relación con las bases teóricas consideradas.

2.3.1 Word2Vec

El análisis semántico del lenguaje no es nuevo y es realizado desde distintas perspectivas. Harris (1954) estudia el lenguaje hablado, donde afirma que la estructura del discurso verbal y la secuencia de enunciados no son arbitrarias, introduciendo además la idea de probabilidad de ocurrencia de una clase de elementos. El autor presenta un elemento que más adelante será retomado por otros en el sentido que dado un enunciado no se puede predecir con exactitud la siguiente palabra. En efecto, el autor sostiene que dado un discurso solamente es posible determinar que tan probable es que el mismo continúe con una palabra en particular. Dicho de otro modo, dado un discurso existirían un conjunto de palabras que continuarán dicho discurso, cada una con una probabilidad de ocurrencia.

No obstante lo anterior, la secuenciación de palabras que dan lugar a una frase no es un elemento exclusivamente probabilístico, sino que también está asociado a las características del discurso, lo que a su vez le confiere sentido semántico (Valle-Lisboa y Mizraji, 2007); siendo adecuado afirmar que el evento probabilístico está (o debería estar) en función de la relación semántica entre los elementos de la clase. Mikolov et al. (2013a) afirman que el análisis semántico de enunciados debería ir más allá de aquel que es atómico, de cada una de las palabras que componen dicho enunciado, presentando además un modelo computacional que resuelve el problema. Es así que en busca de lograr programas más eficientes. Mikolov, Sutskever, Chen, Corrado y Dean (2013b) trabajan con vectores de palabras, en los cuales se busca representar y establecer las relaciones entre las mismas. Los autores observan que considerar un vocabulario amplio es necesario para generar las relaciones semánticas y sintácticas, al tiempo que eliminar del mismo las palabras con una frecuencia menor a 5 aumenta la eficiencia en términos computacionales. El modelo vectorial, donde una palabra se relaciona con otras tantas en función del contexto, se presenta como una buena forma de representar las relaciones sintácticas y semánticas. A modo de ejemplo: la riqueza semántica se visualiza por la posibilidad de inferir que París está relacionado con Francia dado que Madrid los está con España (lo que implica reconocer el hecho que París y Madrid son capitales y que la primer palabra no hace referencia al héroe homérico); o por la posibilidad de diferenciar la cadena “Boston Globe” de la sumatoria de palabras “Boston” y “Globe”. Este tipo de representaciones se ha mostrado eficiente y adecuado, incluso frente a sucesivos ajustes del algoritmo base (Mikolov, Yih y Zweig, 2013).

En nuestro trabajo usamos el algoritmo Word2Vec programado en Python (Řehůřek, 2021) como forma de analizar las relaciones semánticas entre palabras. Word2Vec (Řehůřek, 2021) es un algoritmo de aprendizaje no supervisado, el cual recibe una entrada de datos (textos con sentido, en el cual se establecen relaciones entre palabras), realiza un entrenamiento con los mismos y genera una salida. La entrada de datos son frases con sentido, que representan una realidad, por lo que la relación entre las palabras no es arbitraria, estando el resultado en función de la semántica de dicha realidad. El entrenamiento consiste en establecer las relaciones entre las palabras y la salida una estructura vectorial para cada palabra, donde se cuantifica la relación entre las mismas usadas en el entrenamiento (es posible eliminar del entrenamiento aquellas palabras que tienen una ocurrencia menor de una determinada frecuencia). Las relaciones obtenidas (expresadas mediante valores numéricos en el vector) definen qué tan cercana o lejana está una palabra de otra, lo cual

se asocia a la relación semántica. Para ejecutar el programa debemos definir un texto a usar, entrenarlo y luego seleccionar una palabra para obtener aquellas más cercanas o lejanas a estas. Del mismo modo es posible elegir un vector de palabras, donde el orden no importa, para determinar las más cercanas o lejanas a dicho vector.

Dado que las relaciones que se establecen son semánticas debe tenerse en cuenta que el conjunto de palabras sobre el cual se entrena es de vital importancia, pues es este el que determina la propia semántica de la palabra o conjunto de ellas. En efecto, “rojo” puede ser asociada con “peligro” en un contexto (si el diccionario está asociado a términos químicos, termodinámicos o advertencias de algún tipo), pero se puede relacionar con “color” o “belleza” en otro entorno. Es así que debemos distinguir el uso de esta familia de algoritmos en el contexto del análisis del lenguaje natural, usado comúnmente por los individuos, de su uso en un contexto reducido, especializado, como ser el de un grupo particular de personas. Es el cuerpo textual de dicho contexto, el que determinará la semántica de o las palabras.

Es claro que Word2Vec (Řehůřek, 2021) permite identificar relaciones entre palabras, pero ¿cómo podemos usar el software? A modo de ejemplo consideremos una situación en el cual un estudiante debe verbalizar cómo resuelve ciertas operaciones matemáticas, partiendo del supuesto siempre que el estudiante ha modelado o expresado dichas representaciones en un formato textual. Sumar no es lo mismo que contar, pues el primero implica necesariamente la existencia de valores numéricos, en tanto que el segundo elementos a contar. ¿Sumar, tiene relación con Contar? En un contexto en el cual se expresa que el objetivo es “contar la cantidad de puntos obtenidos”, se puede inferir que el uso de contar es equivalente a sumar. La ejecución del Word2Vec (Řehůřek, 2021) nos podría arrojar luz sobre esta relación entre las palabras usadas.

En todos los casos debemos recordar que la elección de las palabras sobre las cuales realizar la asociación es tal que el experto conoce el contexto, esto es: no debería ser arbitraria ni realizada en función exclusiva del idioma, sino que deberían ser palabras que tienen relación con el problema en cuestión. Este aspecto es de suma importancia, pues como dijimos y ejemplificamos previamente, la connotación de la palabra está relacionada con el contexto en el cual es usada.

En nuestro trabajo usamos el algoritmo descrito con el fin de estudiar la relación entre palabras. De este modo es posible identificar la relación entre una palabra

y el resto del discurso para cada uno de los grupos de estudiantes. Es así que estudiamos qué palabras se relacionan con otra dada, para cada uno de los tres grupos de estudiantes.

2.3.2 K - Means

Los algoritmos de aprendizaje no supervisado son aquellos que dado un conjunto de datos generan una salida, no siendo necesario un modelo previo (o conjunto de ellos) que determine la salida esperada. K - Means es un algoritmo de aprendizaje no supervisado, propuesto por MacQueen (1967), como una forma de agrupar datos n dimensionales, registros, en una cantidad dada de clústers. Cada clúster tiene lo que se denomina centroide, sobre el cual se agrupan el resto de los registros minimizando la distancia entre cada uno de los registros y dicho centroide, el cual se desplaza a medida que el algoritmo itera. Si bien existen trabajos que relacionan la cantidad de elementos a agrupar con la cantidad de clústers (Pham, Dimov y Nguyen, 2005), la determinación de dicha cantidad es difícil, siendo la revisión visual una opción a considerar.

Más allá de la cantidad de clústers elegidos, cada uno agrupa elementos relacionados entre sí en función de los componentes de la dimensión. Se debe tener en cuenta que cada fila de la matriz de dimensión n, da lugar a un registro, debiendo tener relación con las características del elemento a estudiar. La gran fortaleza del algoritmo está dada por la ausencia de un modelo de control, siendo el propio algoritmo el que define los registros que integran cada clúster. El hecho de ser un algoritmo de aprendizaje no supervisado se constituye en una fortaleza, pues el agrupamiento de los datos está dado por características de los mismos, algo que no siempre es observable a simple vista. Este aspecto es particularmente importante cuando cada elemento está definido en función de varias variables, las cuales interactúan para explicarlo pero de una forma no siempre clara para el investigador.

A modo de ejemplo presentamos una par de situaciones posibles en las cuales podríamos hacer uso del algoritmo:

1. Consideremos un conjunto de liceos de los cuales se conoce el grado de repetición de los alumnos, la antigüedad promedio de los docentes y del equipo de dirección y la cantidad de profesores efectivos por concurso. ¿Están relacionados los institutos de algún modo en función de los datos dados? La aplicación de K - Means podría identificar los clústers de liceos, los cuales estarían relacionados entre sí. De este modo se podría

inferir, por ejemplo, que uno de los clústers agrupa a los liceos que tienen docentes con grado 4 o mayor y equipos de dirección con más de 3 años ejerciendo.

2. ¿Podemos establecer agrupamiento de estudiantes en función de los datos socioculturales? Pensemos en el caso de agrupar alumnos en función del grado cultural de los padres, el cual no necesariamente concuerda con el barrio en el cual viven los estudiantes. Parametrizar la relación entre las dos variables anteriores no solo no es sencilla, sino que se estaría definiendo por la vía de los hechos los agrupamientos. El uso del algoritmo K - Means asegura una "objetivación" de las relaciones a través de los clústers.

En nuestro trabajo usamos el algoritmo K - Means provisto por Congacha y Valladoliden (s.f.) su página web para identificar regularidades en función de los agrupamientos generados, procurando verificar si alumnos con respuestas que siguen cierto patrón se agrupan o no en un mismo clúster. El análisis cuantitativo del tipo de respuesta asociada a cada clúster permitiría inferir si dicho clúster agrupa razonablemente bien a estudiantes que cumplen ciertas características, en nuestro caso trabajamos con las variables T, indicando más adelante sobre cuáles hicimos el análisis..

2.3.3 SPEECH GRAPHS

Formalmente un grafo "consiste en un conjunto V de vértices (o nodos) y un conjunto E de aristas (o arcos) tal que cada arista e ∈ E se asocia con un par", donde dicho par puede ser ordenado o no ordenado, haciendo que el grafo también sea o no ordenado (Johnsonbaugh, 2005, p. 320). Dicho de otro modo, un grafo representa relaciones (aristas) entre un conjunto de elementos (nodos), donde la combinación de unos con otros modelan una realidad, asociado esto último al grafo en sí mismo. Un ejemplo clásico del modelado mediante grafos lo constituye un mapa, donde cada ciudad, pueblo o incluso cruce de caminos es representado por un nodo. A su vez, la existencia de una arista que une dos nodos representa una ruta entre ellos, una forma de ir de un lugar a otro del mapa. Cada una de las aristas incluso podría tener un costo asociado, esto es, una forma de representar el ir de un lugar a otro del mapa,

El análisis del discurso mediante el estudio de su grafo representativo no es nuevo, permitiendo identificar las grandes ideas del mismo, en particular frente a la presencia de patologías (Cabana, Valle-Lisboa, Elvevåg y Mizraji, 2011).

Speech Graphs (Mota, Furtado, Maia, Copelli y Ribeiro, 2014) es un software que permite representar un discurso mediante una estructura de grafos, llegando a usarse para realizar análisis del discurso de personas con patologías psiquiátricas (Mota, 2017). El programa parte de un texto, independientemente del lenguaje, elimina caracteres como = , +, etc., y define cada palabra como el nodo de un grafo, siendo las aristas la forma de representar la secuenciación entre una y otra palabra. Conjuntamente con el grafo, que puede o no ser dirigido, el programa presenta una serie de estadísticas.

Speech Graphs no realiza un análisis semántico del enunciado (Simabucuru, Copelli, Ribeiro y Mota, 2020) , sino que representa el mismo con forma de grafo, quedando el primer aspecto mencionado a criterio del investigador. Mota, Sigman, Cecchi et al. (2018) realizan un análisis no semántico basado en grafos de la incidencia de la escolarización en la forma de hablar de las personas, así como la afectación de la psicosis a dicho proceso. Este aspecto es particularmente importante, pues la secuencia “y que hay o no” es representada de igual forma que “hoy es un día lindo”, aunque su significado es claramente distinto. En tal sentido debería considerarse el contexto del discurso si es que la semántica del problema es importante. El software se ha usado con para análisis cognitivos, donde se ha observado que los estudiantes con mayores coeficientes intelectuales y mejores desempeños en lectura tienden a usar más palabras (nodos), conexiones entre ellas (aristas entre otros) y menos repeticiones de las mismas que los estudiantes de bajo rendimiento (Mota et al., 2016)

Algunas de las estadísticas y datos obtenidos por el software son:

1. Nodos: cada nodo representa una palabra distinta, por lo que su cantidad sería un indicador de la riqueza del vocabulario del individuo. Pero tamaño de vocabulario no es lo mismo que claridad o riqueza semántica del discurso. Como el software no elimina artículos o incluso palabras mal escritas, la mera cantidad de nodos debería ser analizada en función del contexto. Por tanto la cantidad de palabras del discurso es un dato que se debería relativizar, siendo importante en función del contexto, es decir, mediante la relación semántica que establece con el resto de las palabras (Hinojosa, 2008).

2. Aristas: permiten establecer las relaciones de precedencia entre las palabras, las cuales pueden estar ordenadas o no (grafo dirigido o no dirigido). Análogamente al caso anterior, una mayor cantidad de aristas estaría indicando una mayor relación entre las palabras, un discurso más

extenso, que en situaciones de ausencia de patologías estaría asociada una mayor riqueza semántica.

3. Nodos Hub: estos nodos concentran la mayor cantidad de aristas del grafo, pudiendo verse como aquellos que articulan distintos sub grafos. A nivel macro un discurso, no trivial, está constituido por distintos discursos, es decir, sub grafos. Sin embargo debe tenerse en cuenta que estos nodos serán principalmente artículos o conectores sintácticos, lo que en este caso podría estar distorsionando la interpretación de los datos. El lenguaje escrito obliga a una mayor reflexión que el oral (Rabazo, Moreno y Rabazo, 2008), por lo que la eliminación de artículos, conectores o muletillas permitiría reducir su incidencia, haciendo que el hub cobre una mayor importancia en el discurso.

4. Coeficiente de clusterización: este coeficiente mide qué tan agrupado se encuentra un nodo en relación al resto, en palabras de Watts y Strogatz (1998) “Suppose that a vertex v has k_v neighbours; then at most $k_v(k_v-1)/2$ edges can exist between them (this occurs when every neighbour of v is connected to every other neighbour of v). Let C_v denote the fraction of these allowable edges that actually exist. Define C as the average of C_v over all v” (p. 441). En el caso del análisis de textos, el coeficiente de acoplamiento estaría dando una idea de la relación existente entre las palabras. Un mayor coeficiente de clusterización estaría en consonancia con textos en los cuales las palabras estarían más relacionadas entre si. Inversamente, un coeficiente de clusterización menor estaría mostrando términos menos relacionados semánticamente, generando discursos menos cohesivos.

5. LCC y LSC: estas variables muestran la componente conexa más extensa, para grafos dirigidos y no dirigidos respectivamente. Este grupo de coeficientes estarían dando una idea de relaciones entre conceptos en un mismo discurso, donde cada uno de ellos forma un sub grafo. Es así que estas variables se podrían asociar a enunciados con sub conceptos que se relacionan en el enunciado original.

Speech Graph fue usado para de analizar las relaciones entre palabras usadas (nodos y aristas) y tratar de identificar regularidades en el discurso. En tal sentido realizamos el estudio sobre grupos distintos de enunciados: primero aquellos enunciados completos de cada uno de los alumnos, la respuesta dada a las preguntas realizadas; y segundo, sobre grupos de enunciados que son un subconjunto del anterior, como en el caso de los enunciados lematizados.

Capítulo 3

Resultados obtenidos

3.1 Grupo de datos analizados

Dado que nuestra intención ha sido determinar cómo es el proceso de modelado de un programa por parte de adolescentes que cursan Ciclo básico de enseñanza secundaria, es que consideramos apropiado relacionar estas capacidades de modelado con los resultados obtenidos en la construcción del programa. En este caso partimos de la base que existe relación entre la correctitud del modelado y la construcción del programa asociado a ella.

Para el análisis de datos consideramos la calificación obtenida por los alumnos (en todos los casos se redondeó la calificación). En algunos casos los estudiantes fueron agrupados en función de las calificaciones obtenidas en la construcción del programa solicitado y que dieron lugar a las siguientes variables:

1. ALTA_N: son aquellos estudiantes que obtuvieron una calificación entre 12 y 9 en la construcción del programa.

2. MEDIA_N: está compuesto por los estudiantes que han obtenido una calificación entre 8 y 5 en el programa.

3. BAJA_N: son los estudiantes que han obtenido 4 o menos de nota.

Ahora bien, el agrupamiento anterior en función de los resultados obtenidos en la construcción del programa no determina que el análisis de datos tenga en cuenta dichas categorías. Dicho de otro modo, estas categorías podrían ser consideradas, pero no necesariamente en todos los análisis realizados. Por lo expuesto corresponde hacer las siguientes observaciones en relación a los criterios usados:

1. En algunos casos consideramos los datos obtenidos de todos los alumnos. Este análisis nos permite identificar las posibles relaciones existentes en general, permitiendo estudiar la globalidad de los datos, para luego determinar si los mismos se ajustan a un determinado perfil o grupo.

2. En otros casos realizamos el análisis en función de los resultados obtenidos por los estudiantes en los programas. En este caso procedimos a dividir los alumnos en función de los tres grupos indicados, para luego realizar el estudio de cada grupo independientemente. Este criterio fue usado especialmente para analizar la semántica de lo expresado por los estudiantes en las respuestas escritas. Esta forma de trabajo es especialmente importante en el uso de los métodos que estudian relaciones semánticas, pues en caso de no aplicar esta división estaríamos incluyendo en una misma categoría a todos los alumnos, independientemente de cómo es el modelado de los mismos.

Analicemos un poco más las implicancias de cada una de las opciones anteriores.

Si queremos reconocer los conceptos más importantes a los que hacen referencia los estudiantes o las relaciones semánticas que se establecen entre conceptos, deberíamos considerar todos los estudiantes. En este caso el universo a estudiar lo constituyen todos los estudiantes, independientemente de otros resultados obtenidos. Como nuestra intención en este caso sería identificar cómo se comporta el conjunto de todos los estudiantes, un agrupamiento de los mismos no aportaría mayor información.

No obstante lo anterior, en el caso de nuestro trabajo es necesario también considerar criterios asociados al modelado, o un indicador de éxito, relacionado en este caso con la calificación. De este modo podríamos analizar relaciones existentes en función de cómo se ha resuelto el problema. Lo anterior nos estaría permitiendo estudiar las diferencias o similitudes evidenciadas en función de qué tan correcta ha sido la solución al problema, es decir, en función de cada uno de los sub grupos definidos. Por tanto, para algunas actividades de análisis

de datos, procedimos a dividir a los estudiantes en función de la correctitud de la solución realizada.

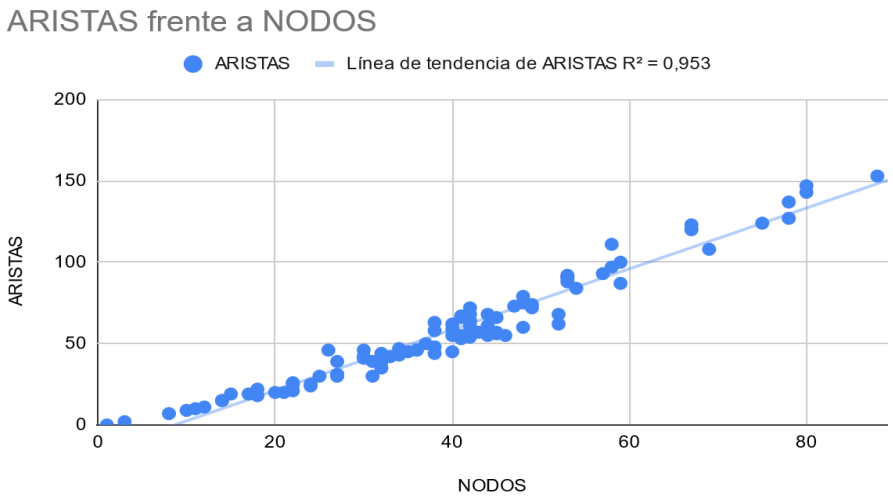
En el presente trabajo aplicamos los dos criterios indicados, explicitándolo en cada caso.

3.2 Análisis del grafo de respuestas

Para las siguientes actividades usamos el programa SpeechGraph con un grafo dirigido. Los datos fueron obtenidos al ejecutar el programa con cada una de las respuestas de los estudiantes a la consigna problema. Una vez logrado lo anterior procedimos a relacionar el resultado con otras variables evaluadas a partir de las respuestas.

3.2.1 Relación entre Nota y Suma_T, Suma_R y Porcentaje de Exceso de Aristas

Primero observamos que existe una relación directa muy fuerte entre la cantidad de nodos (palabras) y aristas (relaciones) en los grafos generados. Como se observa en la Gráfica 1 esta relación es directa y un ajuste lineal tiene un R^2 del 95 %. Este resultado debe ser contextualizado, pero estaría indicando que la cantidad de palabras se asocia también a una mayor verbalización.



Gráfica 1: Relación entre nodos y aristas

Si bien el R^2 anterior es alto, decidimos analizar si la correlación existente entre la cantidad de nodos y aristas fue producto del azar o si bien existe significancia estadística; realizadas las pruebas t y Fischer, obtuvimos correlaciones mayores al 0,97 con $p<0,01$.

Posteriormente procedimos a calcular el porcentaje de exceso de aristas por nodo (porcentaje de aristas por encima de los nodos, ARISTAS % NODOS - 1) para cada estudiante. Este porcentaje (PEA de aquí en más) está indicando cuántas aristas más que nodos tiene el discurso de un estudiante, asociado esto a la construcción de enunciados. Entendemos que esta relación estaría en línea con la riqueza semántica (aunque no necesariamente correctitud disciplinar del discurso), pues es de esperar que aquellos estudiantes que mejor se expresen presenten también más aristas por nodo (esto es, más combinaciones posibles de enunciados). Los discursos de dos alumnos son representados en las figuras 2 y 3, ambas con la misma cantidad de nodos. Figura 2 evidencia una mayor cantidad de aristas, con más relaciones entre las palabras y por tanto un PEA superior a figura 3; y en una primera instancia una mayor cantidad de nodos hub que en el caso de figura 3.

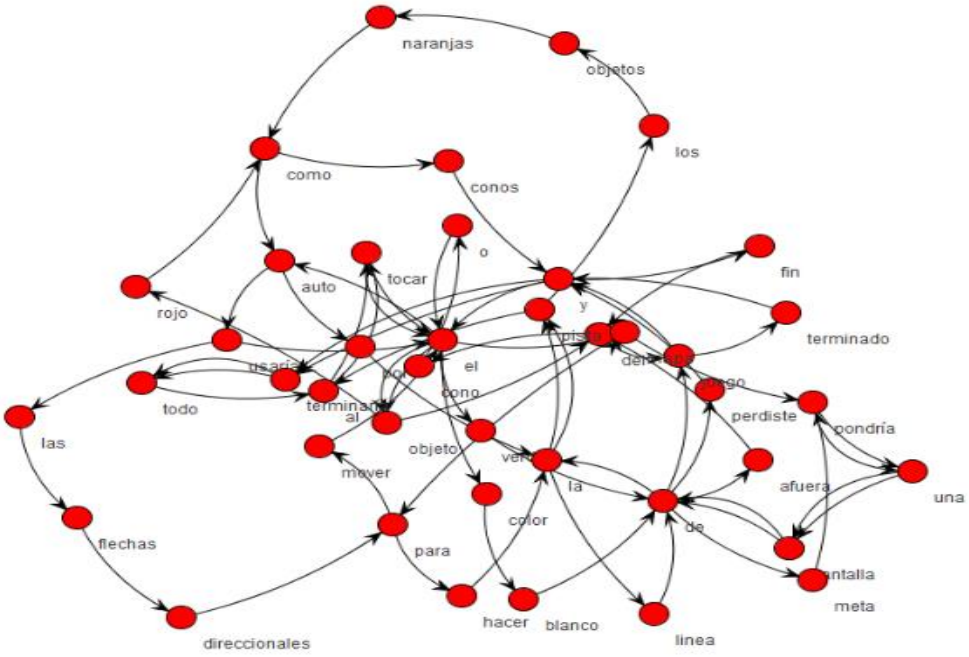


Figura 2: discurso de un estudiante con nota 12, 42 nodos y PEA 71,43%.

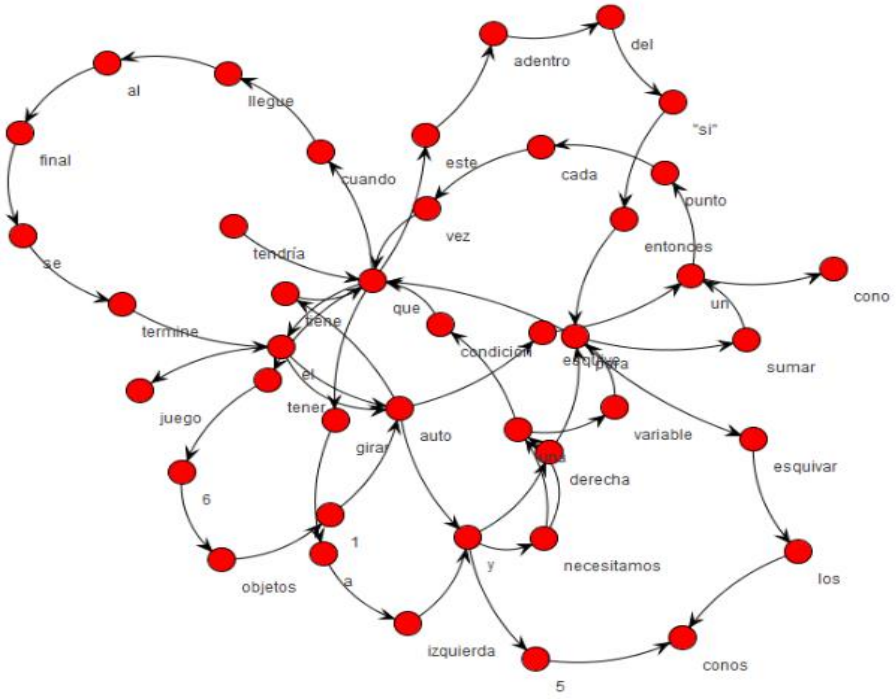


Figura 3: discurso de un estudiante con nota 10, 42 nodos y PEA 28,57%.

Otros dos grafos son presentados en las figuras 4 y 5, uno con PEA de 7,14% y otro con PEA de 62,82%. Podemos observar que el primer grafo denota una linealidad de ideas, con un desarrollo muy pobre y que considerando los nodos involucrados no aborda las dimensiones más importantes del problema. El grafo de la figura 5 muestra una evidente complejidad en la cantidad de nodos, los términos asociadas a ellos y las relaciones que se establecen medidas por la cantidad de aristas. Cuando evaluamos la correctitud del modelado el alumno de la figura 4 obtuvo 0 en todas las dimensiones, en tanto que para el alumno de la figura la evaluación fue 2 para cada una de ellas.

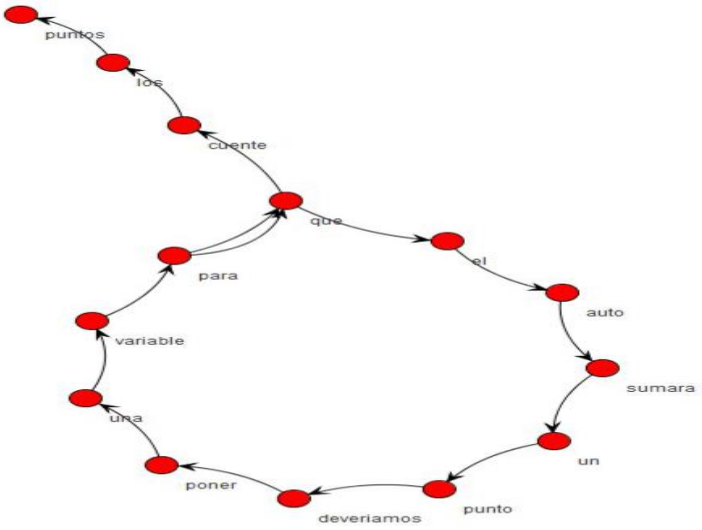


Figura 4: discurso de estudiante con PEA 7,14%

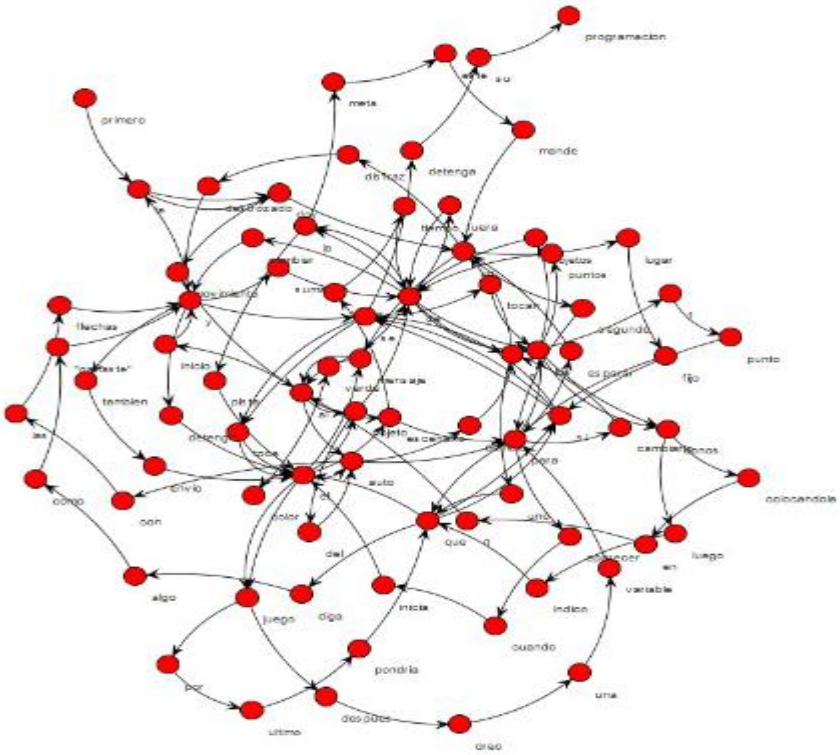
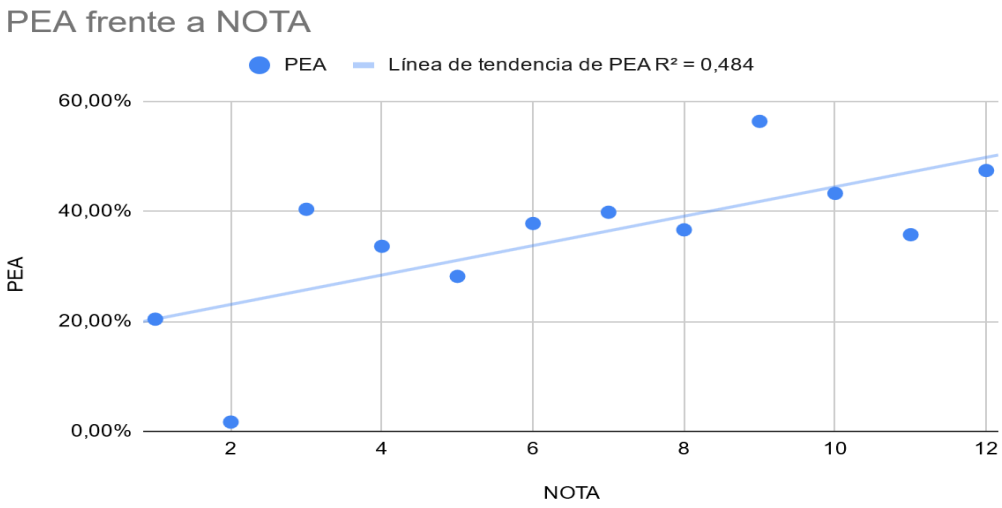


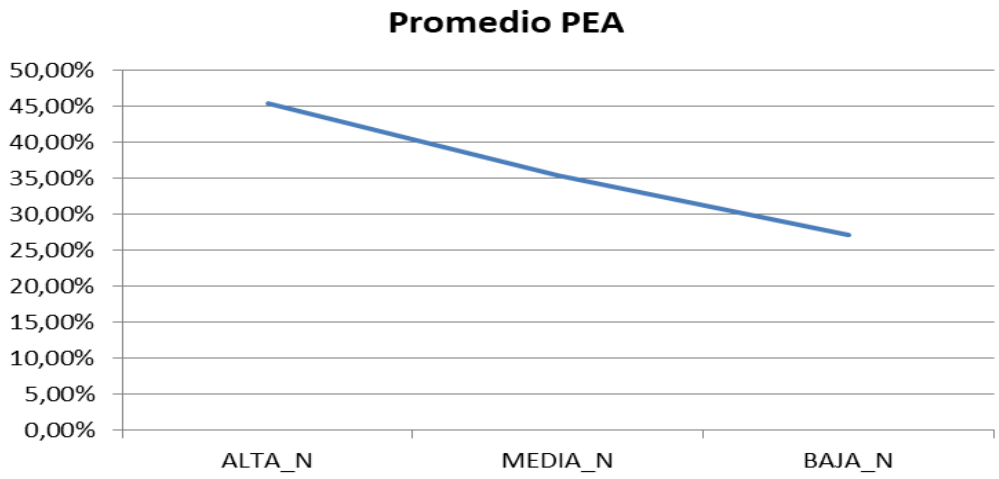
Figura 5: discurso de estudiante con PEA 62,82%

Luego de hacer el cálculo de PEA para cada estudiante, promediamos los valores en función de la calificación obtenida. En una primera instancia la

relación estudiada puede parecer poco clara si consideramos cada calificación. Sin embargo, realizada la gráfica 2 vemos una tendencia positiva en la relación entre la nota y PEA, con un R^2 del 48 %. Al mismo tiempo, al agrupar las calificaciones en tres grupos en función de la misma (ALTA_N, MEDIA_N y BAJA_N), observamos una relación directa entre el grupo de calificaciones y el promedio de PEA. Debemos recordar que la frecuencia obtenida para cada una de las notas no es homogénea, aspecto que podría estar distorsionando los propios porcentajes.



Gráfica 2: Relación nota y promedio de PEA.



Gráfica 3: Promedio PEA según categoría de calificaciones

Vistos los resultados es que procedimos a realizar otro agrupamiento de datos, el cual consistió en definir las tres categorías de calificaciones mencionadas previamente: ALTA_N (entre 12 y 9 inclusive); MEDIA_N (8 a 5 inclusive); y BAJA_N (4 a 1 inclusive). Estos grupos de calificaciones entendemos permiten dar cuenta de tres grupos bien diferenciados de rendimientos, y con casi la misma cantidad de elementos en cada una de las categorías. La gráfica 3 muestra los promedios anteriores solo para el grupo de calificaciones considerados.

En este caso podemos ver que en promedio aquellos estudiantes que han obtenido una mayor calificación se relacionan con una mayor relación entre arista y nodo, lo que podría interpretarse como una mayor riqueza semántica en la verbalización de la solución, en este caso para las categorías de calificaciones consideradas. Sin embargo, cuando consideramos la tendencia que se observa entre cada una de las notas y su PEA correspondiente, no se observa la misma tendencia, siendo el R^2 de 0,07. La cantidad de estudiantes en cada categoría es muy similar (aproximadamente un tercio para cada una), por lo la tendencia es menos sensible a valores atípicos, pareciendo más consistente el resultado.

Considerando todos los estudiantes y el PEA de cada uno de ellos, es que procedimos a realizar el test ANOVA para los tres grupos de calificaciones el cual nos dio un estadístico F de 2,99 con p = 0,0551.

Posteriormente realizamos las correlaciones entre las variables pero en función de la categoría de calificaciones a las que pertenecía (ALTA_N, MEDIA_N y BAJA_N). Los resultados de las correlaciones Pearsons anteriores entre NOTA y las variables, SUMA_T, SUMA_R y PEA han evidenciado ser no significativos al 95% para los test t y Fisher a no ser la correlación entre NOTA y SUMA_T para el sub grupo MEDIA_N. Los resultados completos se encuentran en los anexos.

Ahora bien, la ausencia de una alta correlación puede ser producto de la baja linealidad, algo que podría estar acentuándose en poblaciones relativamente reducidas como son los de los tres grupos de notas. Por tanto procedimos a analizar la correlación entre la variable NOTA obtenida y SUMA_T, SUMA_R y PEA, pero en este caso considerando a toda la población; resultados que se muestran en Tabla 2.

	SUMA_T	SUMA_R	PEA
NOTA	0,555**	0,269**	0,262**

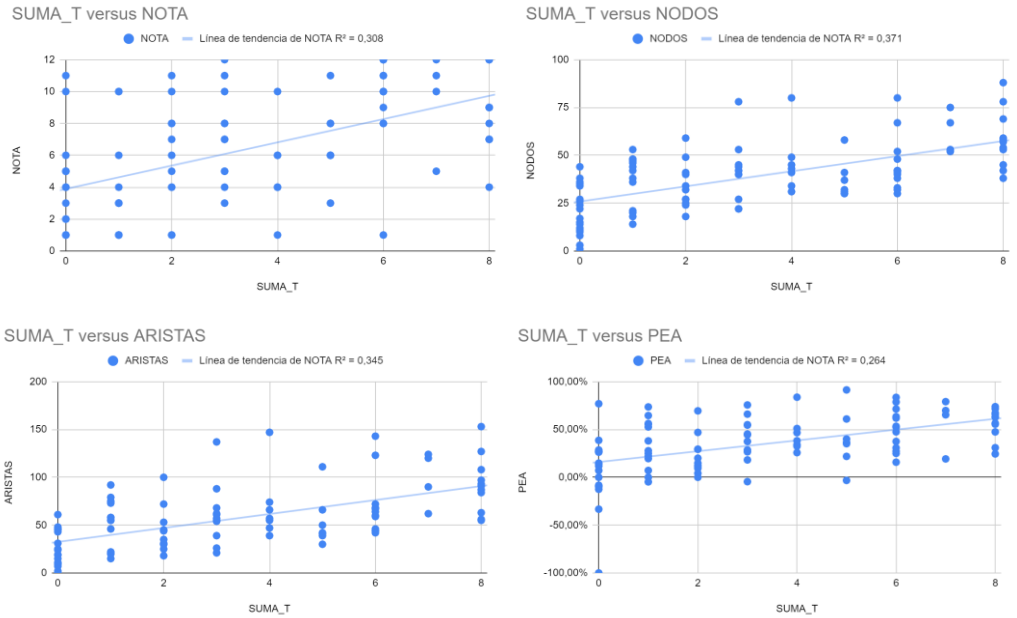
Tabla 2: correlaciones entre la variable NOTA y SUMA_T, SUMA_R y PEA para toda la población. * p al 95%; ** p al 99%.

En primer lugar podemos observar que todas las correlaciones son significativas al 95%, destacando la relación entre NOTA y SUMA_T. Esta modificación en los resultados respecto del agrupamiento según la calificación, posiblemente es producto del hecho que los datos por categoría no llegan a tener la potencia estadística como para detectar los efectos.

3.2.2 Análisis según SUMA_T

Una breve revisión visual de los datos base sobre los cuales trabajamos nos permite determinar que la cantidad de respuestas correctas (y que por tanto han tomado el valor 2 en las variables) tiende a agruparse más frecuentemente con las calificaciones mayores. Por ejemplo: 8 de los 9 estudiantes con variable NOTA 12 han obtenido el valor 2 en T_CONDICIONES, en tanto que solo 1 de 13 estudiantes con calificación 1 ha obtenido 2 en la misma variable. Si consideramos ahora los estudiantes que obtuvieron 0 en la variable mencionada, en el primer grupo de estudiantes ninguno cumple con esta condición, en tanto que en el grupo de quienes obtuvieron 1 en la variable, 6 son los estudiantes. Lo anterior nos llevó a tratar de considerar el problema como un todo, es decir, partimos de la base que la suma de los valores obtenidos en las distintas T_VARIABLES está asociado a la representación global que hace el estudiante del problema a resolver. Dicho de otro modo, asumimos que la variable SUMA_T da cuenta de la visión global de cómo el estudiante se expresa en relación al problema a modelar.

De este modo procedimos a analizar la relación entre las variables del problema. Considerados todos los estudiantes graficamos la relación entre y SUMA_T y las variables NOTA, NODOS, ARISTAS y PEA (Gráficas 4); donde se observan R^2 de 0,308, 0,371, 0,345 y 0,264 respectivamente (las correlaciones han resultado todas significativas al 99%).



Gráficas 4: Relación entre SUMA_T y NOTA, NODOS, ARISTAS y PEA

Las correlaciones entre SUMA_T por un lado y NOTA, NODOS, ARISTAS y PEA son de 0,5548, 0,609, 0,587 y 0,513 respectivamente, todas significas al 95%. Dado que tenemos tres pruebas distintas , una por cada grupo de correlaciones estudiadas (SUMA_T - NOTA, SUMA_T - NODOS, SUMA_T - ARISTAS y SUMA_T - PEA), es que realizamos el ajuste de las pruebas por Bonferroni. Con n = 4 y el nivel de significancia en 5%, el nuevo valor de p pasa a ser 4,9% ($1 - (1 - 0,05 / 6)^6$). De este modo todas las correlaciones mantienen su significancia.

SUMA_T	PROMEDIO PEA	ALTA_N	MEDIA_N	BAJA_N
8 a 6	53,63%	19	7	2
5 a 3	41,69%	4	13	7
2 a 0	19,28%	6	11	25

Tabla 3: cantidad de alumnos y promedio de PEA agrupados por grupo de calificaciones y resultado de SUMA_T. * p al 95%; ** p al 99%.

Posteriormente realizamos un agrupamiento de datos en función de los grupos de calificación definidos. La tabla 3 nos muestra la relación entre el valor obtenido de SUMA_T (agrupados en tres rangos de valores), el promedio de PEA y la cantidad de estudiantes para cada categoría de nota. Si bien podemos observar que tiende a existir una relación directa entre SUMA_T y PROMEDIO PEA, debemos tener en cuenta que el propio promedio podría estar ocultando otras relaciones entre las variables. En tal sentido debemos mencionar que la relación evidenciada no puede ser considerada más que una característica general de parte de la población, sino que esto permite inferir relaciones estadísticas con el resto de las variables. Definir tres grupos de datos en función del resultado de SUMA_T, permite agrupar los estudiantes en función de la correctitud global de la respuesta escrita. Dicho de otro modo, el grupo de estudiantes que tiene SUMA_T entre 0 y 3 estarían mostrando un pobre desarrollo global de la respuestas (más allá de la categoría consideradas); quienes tienen SUMA_T entre 8 y 6 son los alumnos que muestran un alto desarrollo de las ideas evaluadas; los valores de SUMA_T entre 3 y 5 evidencian un desarrollo global de las respuestas intermedio. Es clara y notoria la relación directa entre este último agrupamiento y PROMEDIO PEA. Destaca también el hecho que más del 60% de las calificaciones ALTA_N (12 a 9 inclusive) se concentran en el grupo de estudiantes que tienen SUMA_T mayor o igual a 6. Del mismo modo, más del 70% de las calificaciones entre 1 y 4 se concentran en la categoría de según SUMA_T con valores 2 o menos.

Con los datos de tabla 3 procedimos a realizar el test Chi cuadrado de Pearson, el cual dio como resultado 35,305 con p 4,0206 x 10^{-07} y df 4. En vista del resultado obtenido estamos en condiciones de afirmar que SUMA_T está relacionada con los resultados obtenidos en la construcción del programa, asociado esto a las categorías ALTA_N, MEDIA_N y BAJA_N y que por tanto la relación no es aleatoria. Si bien el agrupamiento de datos en función de las calificaciones se puede considerar arbitrario, observamos que los resultados obtenidos están en función de lo esperado, es decir, quienes tienen un mayor valor de SUMA_T son también quienes en general obtienen mejores calificaciones.

	NODOS	ARISTAS	PEA
ALTA_N	0,641**	0,624**	0,623**
MEDIA_N	0,594**	0,573**	0,499**
BAJA_N	0,529**	0,487**	0,358*

Tabla 4: correlaciones entre la variables SUMA_T y NODOS, ARISTAS y PEA agrupado según rango de calificaciones. * p al 95%; ** p al 99%.

Realizadas las correlaciones entre SUMA_T y las variables NODOS, ARISTAS y PEA, agrupadas por grupos de calificaciones (tabla 4), podemos observar que para las tres variables la correlación es mayor entre quienes obtuvieron calificaciones altas, intermedia en el grupo de estudiantes de MEDIA_N y menores en el caso de los estudiantes que lograron peores resultados. Recordando que la cantidad de nodos está asociada a la cantidad de palabras, la de aristas a la relación que se establece entre dos palabras y PEA al incremento de las primeras en relación a las segundas, podemos decir que quienes han evidenciando mejores soluciones a los problemas son también quienes muestran una mayor relación entre las variables constitutivas del discurso. En relación a la correlación entre NOTA y las variables NODOS, ARISTAS y PEA según grupo de calificaciones, las mismas solamente han resultado significativas para el caso de las notas MEDIA_N. En tal sentido vale mencionar que la ausencia de correlación estadísticamente significativa no implica que la misma no exista, sino que no se puede probar por este medio.

Nuevamente aplicamos Bonferroni, pero esta vez para cada una de las sub categorías, recordando que en este caso agrupamos los datos en función de las calificaciones. Dado que en este caso la calificación (BAJA_N, MEDIA_N y SUMA_N) es un criterio de agrupamiento, el valor de n se mantiene en 3 y con el nivel de significancia original al 5%, el nuevo valor de p pasa a ser 4,39% ($1 - (1 - 0,05 / 3)^3$). Nuevamente todas las correlaciones mencionadas mantienen su significancia.

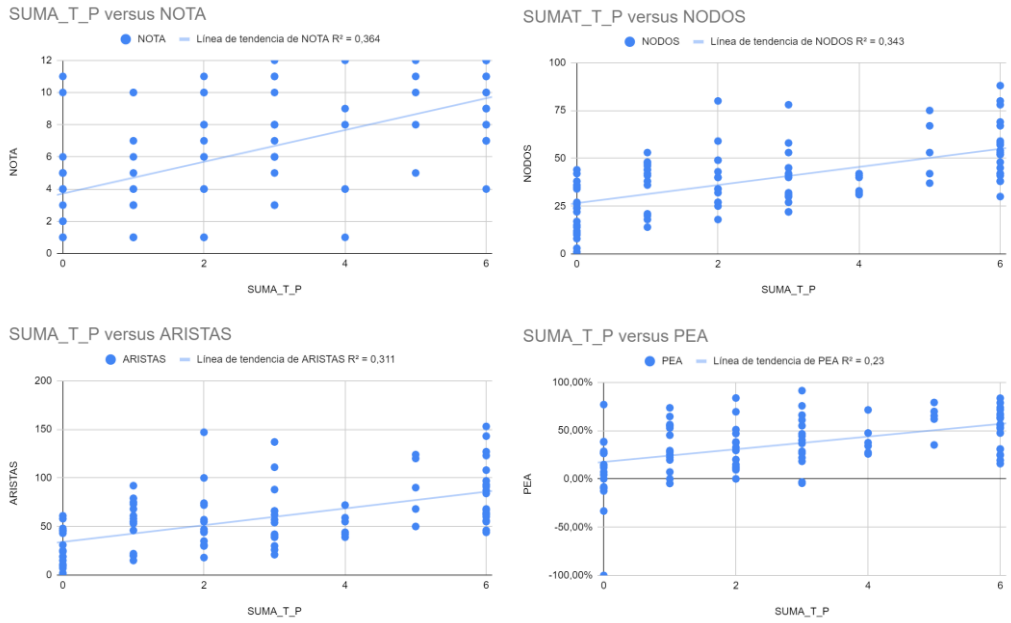
En síntesis y partiendo del supuesto que SUMA_T está asociada a una correcta representación del problema a modelar como un todo y que además integra los principales elementos disciplinares del modelo, podríamos estar afirmando que la calidad de la representación se relaciona fuertemente con una mayor relación

de aristas por nodo y mejores resultados en la programación de la solución. Esta calidad del modelo sería directamente proporcional a la calificación.

3.2.3 Análisis de SUMA_T PARCIAL

Hemos observado que la mayoría de los estudiantes tuvieron un buen desempeño (valor 2) en la variable T_MOVIMIENTOS. Dicha variable cuantifica qué tan bien el estudiante modela el movimiento del vehículo, es decir, establece las relaciones que explican cómo sería la solución. En las actividades de aula el movimiento de los personajes en Scratch tiende a ser una de los ejercicios básicos y fundamentales, teniendo incidencia en casi todas las actividades y desde el principio. Al mismo tiempo, la forma de resolver el movimiento de un objeto puede ser reducida a no más de tres sentencias, lo cual incluso puede ser aprendido cual axioma por parte de los alumnos. Por tanto nos preguntamos si la variable T_MOVIMIENTOS tenía o no relación con los resultados obtenidos. Para ello evaluamos la variable SUMA_T_P (Suma_T parcial), que suma todas las T-VARIABLES con la excepción de T_MOVIMIENTOS.

Realizadas las gráficas de dispersión entre SUMA_T_P y NOTA, NODOS, ARISTAS y PEA (gráficas 5), observamos que el R^2 resultante es de 0,364, 0,343, 0,311 y 0,23 respectivamente. Los valores anteriores son similares a los obtenidos con SUMA_T.



Gráficas 5: Relación entre SUMA_T_P y NOTA, NODOS, ARISTAS y PEA

SUMA_T_P	PROMEDIO PEA	ALTA_N	MEDIA_N	BAJA_N
6 a 5	54,12%	17	7	1
4 a 2	37,28%	9	15	11
1 a 0	19,89%	4	9	22

Tabla 5: Relación entre SUMA_T sin T_MOVIMIENTOS, variables del grafo y cantidad de alumnos.

Con el solo objetivo de mantener frecuencias similares en las tres categorías es que se agruparon los puntajes 6 a 5 en el mismo grupo, de 4 a 2 en otra categoría y la tercera entre 1 y 0, siempre para SUMA_T. A nivel macro se mantienen las relaciones observadas en caso de incluir T_MOVIMIENTOS, donde mayores valores de la variable SUMA_T se asocian a una mayor proporción de PEA.

De 25 estudiantes en la categoría ALTA_N, 17 están comprendidos entre quienes obtuvieron una SUMA_T_P entre 5 y 6, y solo 1 de los estudiantes de este grupo está con calificaciones dentro de BAJA_N. Inversamente, si consideramos los estudiantes cuyo SUMA_T_P está entre 0 y 1, vemos que 22 de ellos tienen calificaciones dentro de BAJA_N y solo 4 con nota dentro de ALTA_N. Los estudiantes que tienen calificaciones MEDIA_N presentan una distribución más uniforme, aspecto que fue visto en todo el trabajo.

El resultado del test Chi cuadrado de Pearsons aplicado a los datos de tabkla 5 nos dio como resultado 31,592, con probabilidad de $2{,}3179 \times 10^{-6}$ y df 4, lo que nos permite afirmar que existe relación entre SUMA_T_P y los grupos de calificaciones tabulados. Estos resultados también son similares a los vistos para el caso de considerar SUMA_T.

A continuación realizamos la correlación entre NOTA y SUMA_T_P, la cual dio como resultado 0,603 con $p < 0{,}01$. Aunque la correlación ha mejorado ligeramente en relación a las observadas entre NOTA y SUMA_T, la mismas no evidencian un cambio fuerte. Cuando se analizan las correlaciones entre

SUMA_T_P y NODOS. ARISTAS y PEA agrupadas según los tres rangos de calificaciones definidos (tabla 6) se observa que mientras algunas correlaciones suben otras disminuyen, en todos los casos ligeramente.

	NODOS	ARISTAS	PEA
ALTA_N	0,643**	0,619**	0,607**
MEDIA_N	0,517**	0,485**	0,399*
BAJA_N	0,522**	0,47**	0,339*

Tabla 6: correlaciones entre la variables SUMA_T_P y NODOS, ARISTAS y PEA agrupado según rango de calificaciones. * p al 95%; ** p al 99%.

Otra vez realizamos el test de Bonferroni con los datos mostrados en la tabla 6, aplicándose los mismos criterios para el agrupamiento de datos que en los casos ya desarrollados. Con tres pruebas (n = 3 y nivel significancia original al 5%) el nuevo valor de p pasa a ser 4,39% ($1 - (1 - 0{,}05 / 3)^3$). Todas las correlaciones mantienen su nivel de significancia a excepción de la correlación entre SUMA_T_P y PEA. No obstante lo anterior, debemos recordar que SUMA_T_P elimina del estudio la variable MOVIMIENTOS_T, siendo una reducción metodológica.

En función de los resultados obtenidos podríamos decir que la inclusión o no de la variable T_MOVIMIENTOS no estaría afectando significativamente los resultados obtenidos. En este caso se observa que los estudiantes con mejores notas en la actividad práctica serían aquellos que expresan mejor la solución, es decir, modelan mejor el problema que quienes se expresan con más limitaciones. Al mismo tiempo y al igual que cuando se estudia SUMA_T en lugar de SUMA_T_P, la relación entre aristas y nodos en el grafo de expresión también tiende a ser mayor entre quienes modelan mejor frente a quienes lo hacen peor.

3.2.4 Análisis según SUMA_R

Otro análisis que realizamos fue en función de SUMA_R, debiendo recordar primero qué representa esta variable. Al principio de las actividades propuestas, les solicitamos a los alumnos que expliquen algunos conceptos fuertes de la programación: estructuras de repetición, condiciones y uso de variables. Cada una de estas preguntas se asoció a una variable que tomó como valor 2, 1 o 0, en función de la correctitud de la respuesta. Al igual que SUMA_T, SUMA_R representa la suma de los valores de las VARIABLE_R para un estudiante dado, lo que podría interpretarse como la correctitud global a una serie de conceptos teóricos. De este modo, SUMA_R podría ser un indicador de qué tan sólidos son los saberes teóricos asociados a la programación.

La división de SUMAR_R en tres categorías sigue los mismos criterios que para las variables ya trabajadas. Sin embargo observamos que las frecuencias son significativamente distintas, por lo que no tenía sentido subdividir cada categoría en función del valor que tomó SUMAR_R. No obstante lo anterior podemos decir que en general existe una relación directa entre SUMA_T y el PROMEDIO PEA como muestra Tabla 7.

		Cantidad de estudiantes por rango de notas		
SUMA_R	PROMEDIO PEA	ALTA_N	MEDIA_N	BAJA_N
6 a 4	50,39%	9	4	5
3 a 2	36,51%	13	13	10
1 a 0	27,62%	8	14	19

Tabla 7: Relación entre SUMA_R, variables del grafo y cantidad de alumnos

El test Chi cuadrado Pearsons dio como resultado 7,063, con p 0,1326 y df 4, por lo que las variables analizadas no están relacionadas y son independientes.

Lo primero que observamos es que el porcentaje PEA mantiene una relación creciente con SUMA_R. Una tercera parte de los estudiantes que obtienen valores de entre 6 y 4 también logran calificaciones altas (ALTA_N), valores similares para quienes tienen SUMA_R entre 1 y 0 y menor que los valores intermedios. Observamos también que cuando SUMA_R toma valores entre 0 y 1 casi el 50% de las notas pertenecen a las notas de la categoría BAJA_N. Por

tanto, podemos afirmar que la ocurrencia de valores de la categoría BAJA_N se concentran entre quienes tienen valores de SUMA_R bajos, no pudiendo observar un comportamiento claro para el resto de las categorías.

3.3 ENUNCIADOS, LEMATIZACIÓN Y CONCEPTUALIZACIÓN

En esta instancia procedimos a analizar los grafos no dirigidos resultantes de enunciados asociados al modelado del problema. Un análisis visual del trabajo con grafos dirigidos no aportó mayor información en relación a la topología de los mismos, aunque sí sobre los nodos, aristas y el agrupamiento de los resultados obtenidos en la actividad práctica.

Los nodos resultantes de los grafos hallados se asocian con palabras, en tanto que las aristas lo hacen con relaciones entre ellas. De este modo, un incremento en la cantidad de nodos está en consonancia con una mayor variedad de palabras, en tanto que un mayor número de aristas estaría indicando una mayor riqueza y variabilidad en la construcción de enunciados.

Sin embargo, una gran ocurrencia de nodos (palabras) no implica necesariamente una mayor riqueza semántica, pues un reducido vocabulario podría llevar a aumentar la cantidad de palabras repetidas (y por tanto de nodos) para representar los mismos significados. En paralelo con lo anterior, una mayor cantidad de aristas podría estar asociado a muletillas, enunciados débiles semánticamente sin coherencia interna o un discurso que no resulte adecuado para la consigna. Partiendo del supuesto que en todo discurso existe un conjunto de palabras que ayudan a comprender el mismo, siendo posible su agrupamiento en grandes familias, es que procedimos a lematizar las respuesta de los alumnos al problema propuesto del modelado (agrupando las palabras bajo sus distintas formas), previa eliminación de las llamadas “stop words” (palabras que acompañan a otras, pero que carecen de un significado autónomo, como preposiciones, pronombres y artículos).

Posteriormente analizamos el discurso en sí mismo en función de la solución modelada. En tal sentido consideramos aquellos elementos y conceptos que entendemos disciplinariamente relevantes para resolver el problema, los cuales van en línea a lo expresado por un grupo de profesores a quienes se les pidió opinión sobre cómo sería una explicación adecuada (Estas opiniones se incluyen en los anexos). Llegamos a la conclusión que existen un conjunto de términos

que son estructurantes del modelado, haciendo a la formas de la solución en sí misma. Identificadas estas palabras (las cuales a su vez se agrupan en familias de ellas en función de su relación con el problema), eliminamos todas las demás, quedándonos con lo que denominamos conceptos fuertes del modelado. De este modo es que podemos construir un enunciado general de la solución, el cual describe una posible solución, la cual a su vez ha sido validada por docentes consultados. Es así que obtenemos el siguiente enunciado incompleto del modelado:

> El **Auto / Coche** debe ser programado para que al **Tocar** un **Cono / Conos / Conito**, que es **Anaranjado / Naranja**, el cual a su vez es un **Objeto / Personaje** del juego, se **Modifique / Cambie** el valor de la **Variable** al Sumar uno al valor total. La Variable **Cuenta / Contaría / Contabiliza** los **Puntos / Puntaje**… Cuando el **Auto / Coche** sale del circuito es porque **Toca / Tocaría / Tocase** el **Color Verde**, **Cambiando / Modificando** el **Disfraz**, lo cual se puede hacer **Desapareciendo / Ocultando / Escondiendo** el **Objeto / Personaje** y **Finalizando / Terminando** el juego.

Obsérvese que un mismo concepto se puede asociar a varias palabras, las cuales permitirían identificar el objeto, relacionarlo con las condiciones de éxito o derrota, finalización o cambio de algún aspecto del juego. De este modo debemos tener en cuenta que la propia definición de los conceptos tiene un grado de subjetividad que solamente puede ser dirimida en función de la realidad a resolver, las características de los alumnos, su lenguaje y la percepción del evaluador. En tal sentido debemos informar que realizamos una corrección ortográfica de los enunciados de los estudiantes, pues muchos de ellos presentaban errores en la escritura como por ejemplo: “obj” en lugar de objeto, “lñea” en lugar de línea, además de otras faltas de ortografía como tildes.

Los conceptos agrupados, y que reiteramos parten de nuestra experiencia y de la de otros profesores, fueron:

Auto / Coche; Cono; Color; Objeto / Personaje; Condición; Variable; Naranja; Verde; Blanco; Linea / Meta; Sumar; Punto; Combinar / Modificar; Tocar; Final / Terminar; Contar / Conteo / Cuenta; Disfraz / Desaparecer / Ocultar / Esconder.

Para este último procesamiento de datos también se consideraron los plurales, tiempos verbales, diminutivos y se corrigieron palabras mal escritas; agrupados los resultados en la variable CONCEPTOS.

A continuación presentamos algunos datos obtenidos mediante SpeechGraph que entendemos merecen un breve análisis recordando que RESPUESTA es la respuesta primaria del estudiante, LEMATIZADA es el resultado de eliminar las stop word y lematizar los enunciado originales, en tanto que CONCEPTOS representa la selección de conceptos fuertes de la respuesta original.

Tabla 8 muestra la cantidad total de palabras, nodos en el grafo, que representan la solución al problema, agrupado en función de la calificación obtenida y categoría de procesamiento. De los datos obtenidos podemos inferir que los estudiantes que logran mejores calificaciones usan más palabras que aquellos que tienen peores calificaciones para todas las categorías, lo que podría estar sugiriendo una mayor riqueza en el uso de lenguaje. Otra interpretación en función de la variable SUMA_T, es el hecho que los estudiantes que tienen mejores notas requieren más y mejores enunciados para representar el problema, mientras que quienes notas más bajas usan menos palabras por el solo hecho de tener una mayor dificultad para modelar el problema.

	GRUPO MODELADO		
CRITERIO	ALTA_N	MEDA_N	BAJA_N
RESPUESTA	2120	1760	1704
LEMATIZADA	1156	938	939
CONCEPTOS	475	389	294

Tabla 8: cantidad de nodos por grupo de enunciados

Resulta interesante la variación porcentual de nodos que se da entre las categorías definidas en función del criterio de respuesta, la cual es mostrada en la gráfica 6. Podemos observar que la cantidad de nodos decrece de forma muy similar en el caso del pasaje de la respuesta original a la respuesta lematizada. Sin embargo, cuando analizamos la cantidad total de nodos resultante de la selección de conceptos fuertes, vemos que existe una reducción mayor entre quienes pertenecen a la categoría BAJA_N. Lo anterior nos estaría pudiendo informar de una menor riqueza semántica y comprensión del problema en función de sus elementos constitutivos.

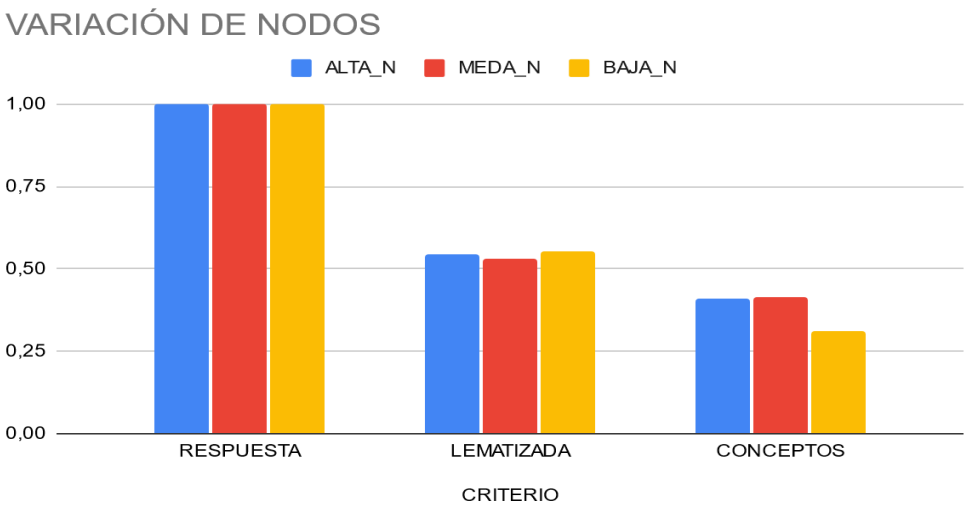


Gráfico 6: Reducción de nodo en función del tipo de respuesta

El siguiente análisis realizado consistió en estudiar la cantidad de nodos, aristas, relación entre aristas y nodos y el coeficiente de clusterización, para cada rango de calificaciones y tipo de grafo. Independientemente de si el grafo estudiado es o no dirigido, se puede ver que la cantidad de nodos, aristas y la relación entre estos, es siempre directamente proporcional a la calificación obtenida para las respuestas originales, lematizadas o de conceptos fuertes.

Para cada una de las categorías de enunciados calculamos la cantidad de nodos, aristas, relación arista nodos y coeficiente de clusterización promedio, datos que a su vez se agruparon en función de la calificación obtenida. Los datos resultantes se pueden ver en Tabla 9, Tabla 10 y Tabla 11.

Lo primero que destaca es que en todos los casos la cantidad de nodos promedio siempre es mayor en ALTA_N que en MEDIA_N y que a su vez en BAJA_N, aspecto válido también para las categorías de respuesta completa, lematizada y de conceptos fuertes. Los mismos resultados se obtuvieron en relación a las aristas y el porcentaje de incremento de aristas respecto de los nodos . La cantidad de nodos nos estarían indicando un vocabulario más variado entre quienes tienen calificaciones más altas, que es directamente proporcional a ella, en tanto que la cantidad de aristas evidencia una mayor relación entre las palabras, con enunciados más complejos y que involucran más palabras. Es así que se estaría observando una mayor riqueza semántica y relaciones más

complejas entre las palabras entre quienes tienen notas más altas que entre quienes tienen calificaciones intermedias y con quienes tienen notas más bajas.

Dado el coeficiente de clusterización vemos que las diferencias son bajas dentro de cada categoría de calificaciones. En el caso particular de los conceptos fuertes (tabla 11) se observa que dicho coeficiente es más alto para las calificaciones bajas en el grafo de conceptos fuertes en relación con quienes tienen calificaciones más altas. Puesto que el coeficiente de clusterización cuantifica la relación entre un nodo y el resto del grafo, el análisis de este indicador estaría dando una idea de la cohesión del discurso. En el caso de los grafos de conceptos fuertes, el incremento del coeficiente seguramente se relacione con enunciados empobrecidos, redundante en términos de palabras, el cual implica por tanto el uso recurrente de los mismos conceptos, haciendo que los mismo se integren más fuertemente con el resto. Debemos recordar que el conteo de nodos es independiente del reconocimiento de sinónimos. Sin embargo, en el caso de los conceptos fuertes, cada nodo representa un concepto necesario en la solución del problema, por lo que desde el punto de vista de la semántica del problema podríamos estar asumiendo que cada nodo agrupa una serie de sinónimos.

	RESPUESTA COMPLETA			
	NODOS	ARISTAS	PEA	CC
ALTA_N	45	62	32%	0,05
MEDIA_N	39	50	23%	0,04
BAJA_N	37	47	22%	0,03

Tabla 9: Estadísticas promedio de grafo para respuesta completa

	RESPUESTA LEMATIZADA			
	NODOS	ARISTAS	PEA	CC
ALTA_N	28	35	19%	0,04
MEDIA_N	24	29	16%	0,04
BAJA_N	22	26	11%	0,04

Tabla 10: Estadísticas promedio de grafo para respuesta lematizada

	RESPUESTA CONCEPTOS FUERTES			
	NODOS	ARISTAS	PEA	CC
ALTA_N	10	12	15%	0,16
MEDIA_N	8	9	3%	0,17
BAJA_N	6	7	1%	0,18

Tabla 11: Estadísticas promedio de grafo para respuesta de conceptos fuertes

Si consideramos la cantidad de nodos promedio de los enunciados de conceptos fuertes vemos que los mismos son mayores entre quienes pertenecen a ALTA_N en relación con MEDIA_N y de estos con BAJA_N (características común a los tres grupos de grafos). Desde un punto de vista estrictamente disciplinar la cantidad de nodos nos estaría dando pautas sobre cómo los estudiantes estructuran el discurso en función del modelado de la realidad a programar. Del mismo modo, el cambio de aristas está directamente relacionado con la calificación obtenida por el alumno, siendo la variación mucho mayor que en el caso de los enunciados de la respuesta completa y lematizada. Es así que se observan diferencias en la cantidad de términos / conceptos usados entre los grupos estudiados, lo que estaría indicando formas de modelado más ricas semánticamente y cercanas a la realidad entre quienes pertenecen a ALTA_N respecto de los estudiantes de MEDIA_N y de estos en relación a quienes pertenecen a BAJA_N. El coeficiente de clusterización resulta mayor entre quienes pertenecen a BAJA_N, lo cual podría ser resultado más que de relaciones semánticas, de un grafo con pocos nodos donde la existencia de nodos influye significativamente en el indicador. No obstante lo anterior debemos matizar estos últimos datos y el procedimiento en general, pues el propio tamaño del grafo podría estar dando a entender resultados incorrectos. En efecto, la cantidad de nodos e incluso los criterios de agrupamiento podrían dar lugar a interpretaciones incorrectas, debiendo ser considerados otros posibles criterios de valoración de los resultados.

3.4 CLUSTERIZACIÓN DE DATOS

Los datos que presentamos a continuación muestran los clústers obtenidos al ejecutar el algoritmo K - Means. Para cada uno de los procesos de clusterización procedimos a relacionar el resultado obtenido con las calificaciones de los alumnos. En todos los casos utilizamos como conjunto de datos la población completa, independientemente de la calificación obtenida u otros criterios. Una vez identificados los registros que conforman cada clúster, procedimos a relacionarlos con el resto de los datos recolectados, en particular con la calificación. En tal sentido recordemos que el rango de calificaciones ALTA_N está comprendido entre la nota 12 y 9, MEDIA_N entre 8 y 5 y BAJA_N 4 o menos.

La cantidad de clústers se definió en función de los resultados obtenidos, pues una cantidad menor no permitía identificar claramente criterios de agrupamiento de datos. En efecto, el algoritmo se corrió varias veces, pero recién cuando definimos 5 clusters es que pudimos identificar patrones claros en relación a los registros de datos.

Tabla 12 muestra los clústers considerando las variables T_OBJETOS, T_VARIABLES, T_CONDICIONES y T_MOVIMIENTOS, recordando que las mismas son el resultado de evaluar la respuesta del alumno a cómo se resolvería el problema según esas cuatro dimensiones. Al mismo tiempo debemos tener en cuenta que para este procesamiento de datos trabajamos con todos los registros obtenidos. En principio destacan los clústers 1 y 3, el primero por concentrar más del 50% de las calificaciones altas y el segundo por hacerlo con más del 60% de las calificaciones bajas, en ambos casos de la población estudiada. En el primer clúster podemos ver que solo 1 estudiante de 24 tiene una calificación baja, en tanto que en el clúster 3 solo 1/3 parte de los estudiantes obtuvieron calificaciones medias o altas. El otro elemento a considerar es el promedio de SUMA_T (redondeado) para cada uno de los clústers. Recordando que el mayor valor posible es 8, observamos que para el clúster 1 el promedio de SUMA_T es 7, en tanto que para el clúster 3 su promedio es 0. Los otros tres clústers presentan pocos registros de análisis en comparación con los anteriores y una distribución relativamente uniforme de las calificaciones obtenidas así como valores relativamente similares de SUMA_T. Dicho de otro modo, una autoorganización de los registros en función de las cuatro variables, agrupa en distintos clústers a alumnos que mayoritariamente tienen buenos resultados en la programación y en la explicación de la solución,

por un lado, y a quienes tienen bajos resultados en programación y en la explicación por otro.

T_OBJETOS, T_VARIABLES, T_CONDICIONES, T_MOVIMIENTOS					
RANGO CALIFICACIONES	C_1	C_2	C_3	C_4	C_5
ALTA_N	16	1	4	4	5
MEDIA_N	7	4	7	5	8
BAJA_N	1	4	21	2	6
PROMEDIO SUMA_T	7	3	0	5	3

Tabla 12: clusterización de datos según 4 variables

Hemos visto que en general son muchos los estudiantes que han tenido una buena respuesta a T_MOVIMIENTOS, esto es, han explicado de forma adecuada cómo es el modelado del movimiento de los objetos involucrados en el programa. Lo anterior puede ser consecuencia de un uso temprano de estas sentencias en el curso, de una mayor cantidad de ejercicios resueltos que hacen uso de las mismas en comparación con los otros contenidos o de formas de solución relativamente simples. En tal sentido nos preguntamos si esta dimensión tenía incidencia en la definición de los clústers, procediendo a eliminar de los entrenamientos esta variable. La tabla 13 muestra el resultado de este nuevo proceso de clusterización. Podemos ver que el clúster 1 no ha cambiado significativamente los valores obtenidos asociados, tanto que el 5 es en esta instancia el que concentra a los estudiantes que han obtenido peores calificaciones, con valores similares a los anteriores. Dicho de otro modo, este nuevo proceso de clusterización muestra resultados similares a los anteriores, en el sentido que dos clusters agrupan mayoritariamente a quienes tienen las mejores calificaciones y representaciones escritas de la solución, por un lado, y a quienes tienen las peores calificaciones y representaciones por el otro. Se observan mismos resultados en el grupo de clusters intermedios.

T_OBJETOS, T_VARIABLES, T_CONDICIONES					
RANGO CALIFICACIONES	C_1	C_2	C_3	C_4	C_5
ALTA_N	17	3	4	2	4
MEDIA_N	7	6	5	5	8
BAJA_N	1	1	7	6	19
PROMEDIO SUMA_T	6	3	2	3	0

Tabla 13: clusterización sin T_MOVIMIENTOS

A continuación nos preguntamos si las respuestas asociadas a los conceptos teóricos, que realizamos previo a la construcción del programa, generaban agrupamientos de registros, y que se pueden ver en la tabla 14. En este caso observamos que los clústers obtenidos no son claros en el sentido de relacionar estudiantes con grupos de calificaciones similares. En efecto, si bien el clúster 2 agrupa casi la mitad de todos los estudiantes que obtienen bajas calificaciones, lo cierto es que también concentra un 30% de los alumnos con calificaciones medias y la cantidad de alumnos con calificaciones altas no difiere significativamente de las obtenidas en el clúster 1 y 5. Así como no podemos decir que el clúster 2 se asocia a un grupo de calificaciones, pues incluye varios estudiantes con otras calificaciones, lo mismo es válido para los otros clústers. No obstante lo anterior destaca el bajo valor del promedio de SUMA_R asociado al clúster 2. De este modo podríamos estar afirmando que las respuestas a las preguntas teóricas no estarían agrupando elementos de modo que estos se relacionen con el resultado de la programación.

R_VARIABLE, R_CONDICIÓN y R_REPETICIÓN					
RANGO CALIFICACIONES	C_1	C_2	C_3	C_4	C_5
ALTA_N	9	7	3	5	6
MEDIA_N	4	11	2	4	10
BAJA_N	6	15	2	5	6
PROMEDIO SUMA_R	2	0	3	3	3

Tabla 14: clusterización en función de variables R

Si bien realizamos otros análisis, los cuales se incluyen en los anexos, no hemos constatado la existencia de patrones claros de agrupamiento en los mismos.

3.5 ANÁLISIS DE RELACIONES SEMÁNTICAS

El último grupo de análisis realizado se basó en el uso del algoritmo Word2Vec, el cual relaciona semánticamente palabras. Los gráficos que mostramos a continuación parten del entrenamiento realizado mediante Word2Vec y dada una palabra muestran aquellas más y menos relacionadas con ella. Para realizar los entrenamientos usamos las respuestas de los alumnos, que son las que determinan las relaciones entre las palabras, las cuales obtuvimos de las respuestas que dieron lugar a: R_Variables, R_Repetición y R_Condición, y la respuesta a la pregunta “Cuéntale a alguien cómo resolverías el programa siguiente:” que dio lugar a las variables T. Al mismo tiempo agrupamos las respuestas en función de la calificación obtenida en la construcción del programa, es decir, en función de los grupos ALTA_N, MEDIA_N y BAJA_N. Recordemos que el criterio usado se sustenta en el hecho que para entender cómo es que modela un determinado grupo de estudiantes, es necesario considerar solamente las respuestas de dicho grupo. En todos los casos, cuando hacemos referencia a la relación entre palabras, debe entenderse que la misma es a ella o a las que tienen la misma raíz.

Para el análisis consideramos las palabras: Punto, Variable, Tocar, Suma, Cono, Color, Condición y Auto, las cuales están asociadas a la solución del problema. En efecto, si pensamos en cómo sería la explicación de dicha solución, estaríamos redactando que un auto (u objeto) debe ser programado para moverse con el teclado o el mouse, a la vez que se suman puntos si el vehículo toca un cono (o el color anaranjado), existiendo condiciones de victoria o derrota en función de si se toca un color u otro objeto. Un ejemplo de redacción que incluye palabras agrupadas por conceptos similares y que resuelven el problema fue explicitado en el apartado Enunciados, lematización y conceptualización.

Un primer grupo de gráficos (gráfico 7) muestra la distribución de palabras relacionadas con el concepto Punto, el cual a su vez se relaciona al manejo de las variables, conos y operaciones aritméticas de acumulación. La primera gráfica, propia de quienes tienen un modelado ALTA_N, muestra precisamente la relación anterior, manteniendo cerca de Punto los términos mencionados. Resultados similares se observan para los alumnos que se encuentran en la categoría MEDIA_N, a no ser por el término Variable, el cual no ha aparecido en los entrenamientos realizados. En el caso de los estudiantes de BAJA_N, podemos ver que los términos obtenidos no tienen relación computacional con el término Punto.

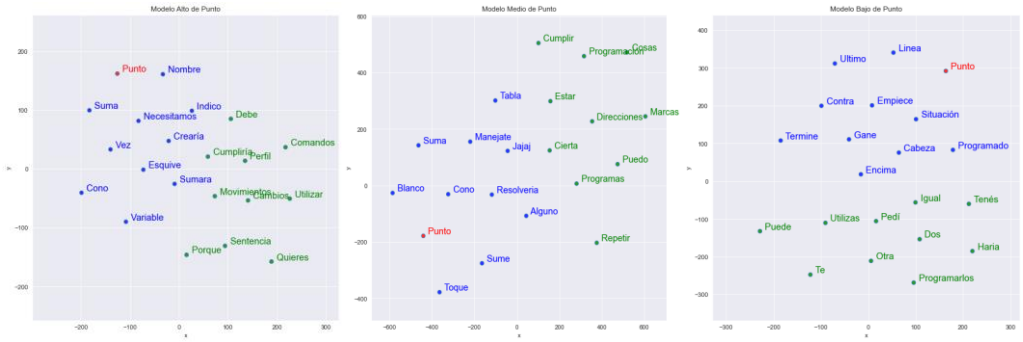


Gráfico 7: palabras más y menos relacionadas con la palabra punto para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

Realizamos el mismo análisis para el término Variable como se muestra en el gráfico 8, el cual se relaciona con Punto, Suma, Cono (en un nivel concreto y propio de la resolución del problema) o Crear (relacionado con la creación de la variable) para los tres grupos de estudiantes. El grupo de estudiantes de modelado ALTA_N presenta nuevamente una fuerte relación con términos como Punto, Crear o Suma (o Cuenta), como era de esperarse. Los estudiantes que tienen un nivel de modelado MEDIA_N presentan resultados similares, aunque existe más palabras que incluso considerándose relacionadas con la redacción (Utilizar o Cuando), no parecen ser adecuadas en función del modelado del problema. El tercer grupo de estudiantes está compuesto por los estudiantes del grupo BAJA_N, en el cual se observa que las palabras mostradas como relacionadas no son significativas para modelar el problema, a no ser por el término Crear.

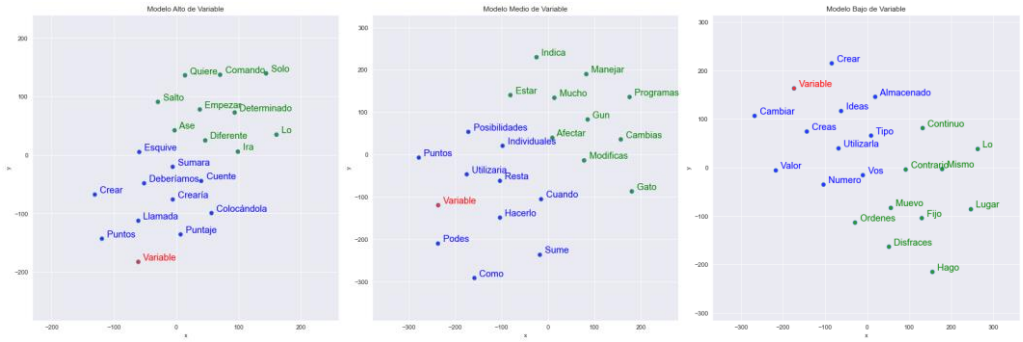


Gráfico 8: palabras más y menos relacionadas con la palabra variable para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

Gráfico 9 muestra nuevamente que quienes modelan de forma adecuada el problema tienen más palabras relacionadas con el término Tocar (Final, Verde o Cambiaría) que quienes realizan un modelado bajo, quienes presentan la palabra Termine como la única relevante desde el modelado del problema. En efecto, los estudiantes del grupo ALTA_N tienes asociados varios términos que dan cuenta de condiciones de finalización o cambio del disfraz del objeto, cantidad que es mayor que en aquellos estudiantes que pertenecen al grupo BAJA_N. En el ejemplo presentado, los estudiantes del grupo MEDIA_N muestran varios términos relacionados (Suma, Naranja, Línea, escrito como Liña o Llegada), aunque debemos recordar que la cantidad y tipo de relaciones entre los términos es sensible al entrenamiento, pudiendo aparecer pequeñas diferencias.

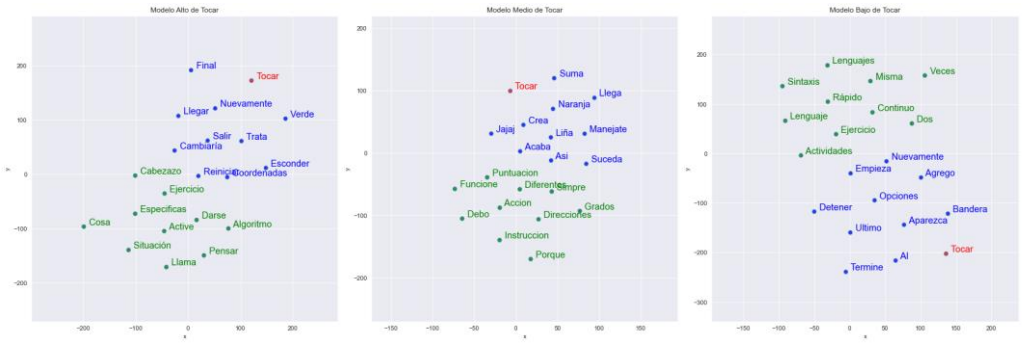


Gráfico 9: palabras más y menos relacionadas con la palabra tocar para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

.

En el caso del término Suma, los resultados son similares para los tres grupos como se muestra en gráfico 10, pudiendo observar que tanto Punto como Cono se relacionan con dicha palabra. En tal sentido se podría afirmar que todos los alumnos identifican la relación entre los conos, la asignación de puntos y el hecho que deben ser sumados o acumulados de alguna forma. Se observa que para los alumnos de MEDIA_N y BAJA_N existen otros términos que no tienen una relación clara con Suma (como el color o aspectos relacionados con la finalización del juego). Visto este último resultado podemos decir que los alumnos que pertenecen a MEDIA_N y BAJA_N logran crear un espacio semántico que capta la idea detrás del término Suma, pero que al mismo tiempo incluye otra tanta información no relevante y que podría poner en tela de juicio la coherencia general de dicho espacio semántico.

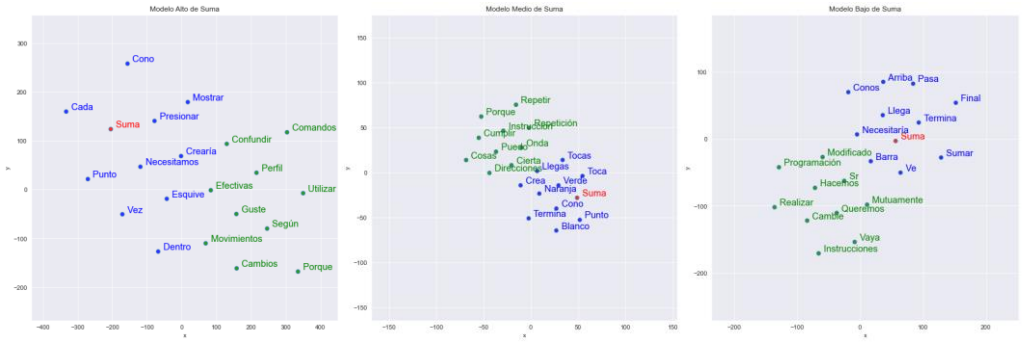


Gráfico 10: palabras más y menos relacionadas con la palabra suma para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

En el gráfico 11 podemos ver que para los estudiantes que tienen un modelado alto el término Cono se relaciona con Punto o Suma, lo que a su vez es consistente con lo expresado en el párrafo anterior. En efecto, mientras Suma se relaciona con Cono y Punto, Cono se relaciona también con Suma y Punto; lo mismo sucede con quienes tienen un modelado Medio, y con los términos Cono y Suma para quienes presentan un modelado Bajo. En este caso se puede afirmar que los estudiantes han relacionado de forma adecuada las palabras indicadas, sea por la comprensión de la consigna, sea por lo simple de la misma.

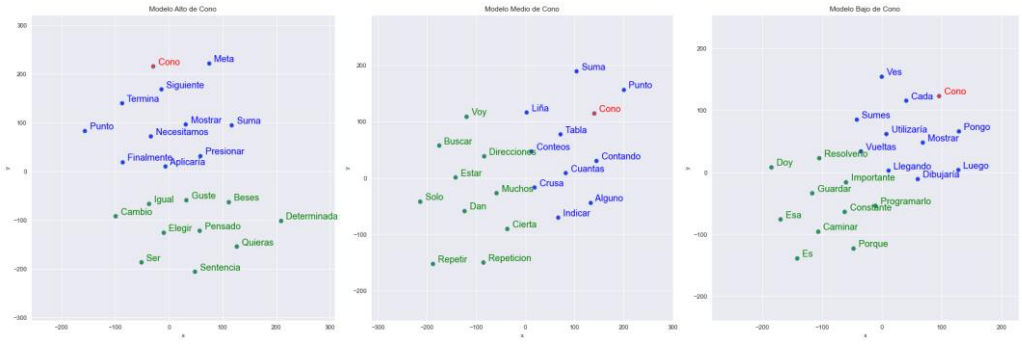


Gráfico 11: palabras más y menos relacionadas con la palabra cono para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

Como podemos ver en gráfico 12, el concepto Color se relaciona con condiciones, en el sentido amplio del término pues al tocarse un color (u objeto)

el juego finaliza, se suman puntos o cambia el disfraz del vehículo (pues este sale del circuito y debe explotar). Los conceptos anteriores se encuentran entre quienes forman parte de los grupos ALTA_N y MEDIA_N. Sin embargo, en el caso de los estudiantes que tienen un bajo modelado la relación entre Color y estos conceptos fuertes no es tan clara. De hecho, para este último grupo de alumnos las relaciones no son claras, incluso si realizamos varios entrenamientos distintos. Para quienes tienen un modelado Bajo el término Color no solo no estaría asociado a conceptos fuertes de la solución del programa, sino que las relaciones que se estaría estableciendo con otros conceptos sería débil o sin un patrón claro.

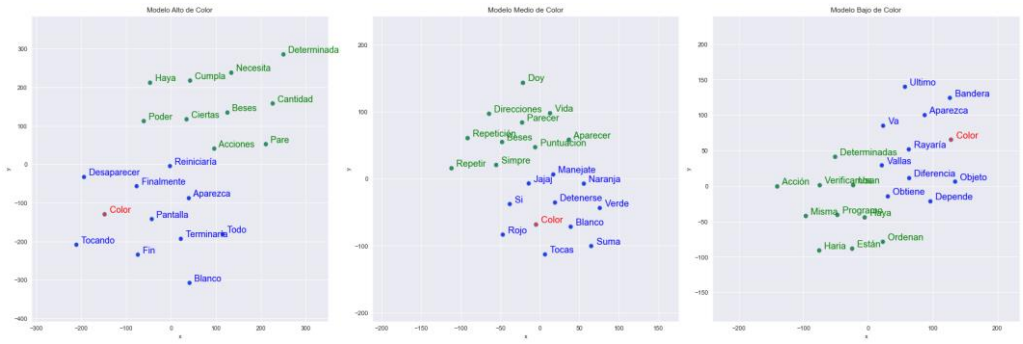


Gráfico 12: palabras más y menos relacionadas con la palabra color para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

Tanto para las palabras Condición como Auto, presentados en gráfico 13 y 14 respectivamente, podemos ver que los términos relacionados a estos se asocian principalmente a acciones propias del movimiento del vehículo, más que a conceptos fuertes de programación. En efecto, palabras como Hacer, Esquivar, Mover o Tecla son recurrentes, aunque más frecuentes en quienes presentan un modelado ALTA_N y MEDIA_N que entre quienes están en el grupo BAJA_N. Estas palabras muestran un alto grado de concretitud de las solución.

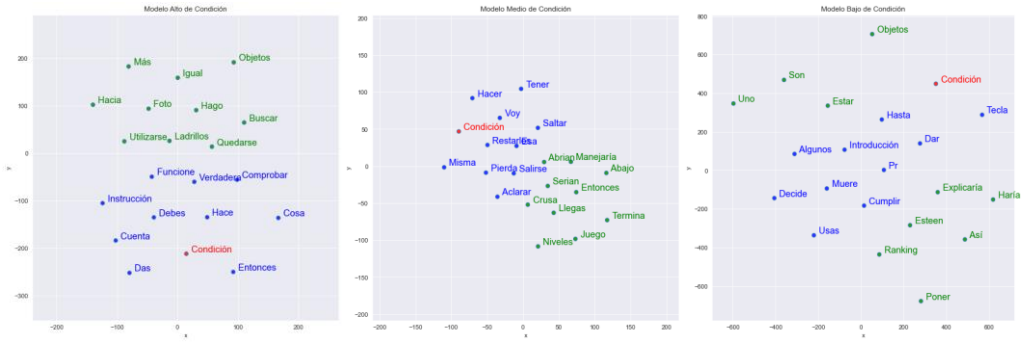


Gráfico 13: palabras más y menos relacionadas con la palabra condición para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

No obstante lo anterior, destaca el hecho que en el caso de Auto hay otros términos propios de la programación (como Cono, Tocar o Terminar), como si el mismo fuese articulador del discurso, relacionando de forma coherente el movimiento con estos conceptos fuertes. Nuevamente, la cantidad y calidad de palabras relacionadas entre quienes pertenecen al grupo ALTA_N es superior a quienes pertenecen al grupo BAJA_N.

Gráfico 14: palabras más y menos relacionadas con la palabra auto para los sub grupos ALTA_N, MEDIA_N y BAJA_N respectivamente.

.

Capítulo 4

Discusión

Un enunciado asociado a la explicación de un procedimiento o concepto puede ser una sucesión de palabras con una baja o alta relación con otras ya referenciadas, lo que dará sentido y coherencia a la explicación. En caso de existir una fuerte relación entre dichas palabras, el modelo con forma de grafo presentará una mayor relación entre aristas y nodos. Hemos observado que existe una tendencia a que los alumnos que han obtenido mejores calificaciones son también quienes en el grafo evidencian una mayor relación entre las palabras. Dicho de otro modo, los enunciados de estos alumnos tienden a ser más complejos, con mayores referencias a palabras ya mencionadas (unidos esto a PEA), más palabras, en particular aquellas asociadas a conceptos fuertes de la solución a programar.

El ANOVA realizado sobre los grupos de calificaciones muestra un p - value apenas superior al 0,05. El dato anterior si bien no modifica sustancialmente la probabilidad obtenida, sí podría estar estar asociado a la falta de potencia estadística de los experimentos realizados, aspecto que debería ser considerado en futuros trabajos.

Otro aspecto a considerar es la coherencia en la explicación de cada una de las ideas fuertes que estructuran la solución del problema. Para cada variable que permite cuantificar SUMA_T hemos observado que las mismas en general toman valor 2 (y por tanto SUMA_T se acerca a 8) para aquellos alumnos que han obtenido calificaciones altas en la construcción del programa. Este valor se

asigna a enunciados que no solo representan de forma adecuada el concepto a evaluar, sino que además el mismo es lo suficientemente coherente como para ser considerado correcto. Si bien en nuestro trabajo no hemos podido usar algún software que cuantifique y de algún modo objetiviza la cohesión del texto, podemos afirmar que las respuestas presentan indicadores de cohesión adecuados, expresado esto por la coherencia, pertinencia y adecuación de los mismos. En efecto, los estudiantes que tienen los mayores valores de SUMA_T son también quienes expresan mejor el modelo, existiendo una superposición de ideas entre la representación del modelo y la realidad dada (Crossley et al., 2019). La evaluación de las respuestas de los estudiantes, al igual que las propias categorías a corregir, entendemos es una debilidad de nuestro trabajo, pues la gran cantidad de enunciados a evaluar y su interpretación podría dar lugar a criterios distintos por parte de cada lector.

Cuando consideramos el discurso de los estudiantes como un todo, visto a través de SUMA_T, podemos ver que los valores más altos de la variable tienen un mayor número de estudiantes con calificaciones entre 9 y 12 inclusive, con un mayor porcentaje de relación entre aristas y nodos. Inversamente, estudiantes con menores valores de SUMA_T presentan también una menor relación entre nodos y aristas, al tiempo que agrupan a una mayor cantidad de alumnos con bajos rendimientos en la actividad práctica. El discurso de los estudiantes (SUMA_T) se relaciona con los resultados obtenidos medidos a través de las calificaciones, no siendo esta relación al azar. Existe correlación estadísticamente significativa (incluso realizando el ajuste por Bonferroni, como en otras correlaciones) y superior al 50% en todos los casos entre SUMA_T y NODOS, ARISTAS y PEA. Lo antedicho se mantiene cuando las correlaciones se calculan en función de los grupos de calificaciones obtenidas. Al observar el comportamiento de los datos en función de las calificaciones, vemos que a mayores notas, también mayor es la correlación, aspecto que estaría indicando que quienes tienen mejores notas también tienen mejores discursos desde lo disciplinariamente, además de más complejos y elaborados.

Si bien la relación entre NOTA y PEA es baja, no sucede lo mismo entre SUMA_T por un lado y NODOS, ARISTAS y PEA por otro, existiendo una correlación superior al 50%. Podemos afirmar que aquellos alumnos que representan mejor el problema, construyen discursos donde las ideas se interconectan más fuertemente (figuras 2 a 5) y no solo de forma secuencial como se observa en la figura 4. Si bien los parámetros de los grafos como ser cantidad y tipo de hub, coeficientes de clusterización o loop entre otros, no aportaron información significativa, creemos que en discursos con un reducido tamaño de nodos, el PEA podría ser un buen indicador de relaciones semánticas

La cantidad de nodos, aristas y la relación entre estos últimos, del grafo que representa el discurso de los estudiantes, tiene una asociación positiva con el grupo de calificaciones, evidenciando un discurso con más palabras y relaciones entre ellas. Lo anterior unido a valores más altos de SUMA_T para los estudiantes de calificaciones más altas, estaría en línea con una mayor sofisticación léxica (Skalicky et al., 2017). Al mismo tiempo se ha identificado un conjunto de palabras asociadas a conceptos fuertes que hacen al modelado de la solución, sin las cuales no es posible describir de forma adecuada el problema. Este conjunto de conceptos, aunque reducido, es cuantitativamente mayor entre quienes han obtenido calificaciones altas que en el resto de los estudiantes.

Los valores de SUMA_T sugieren que en general existe una coherencia y cohesión semántica en el grupo de estudiantes de ALTA_N superior al grupo BAJA_N, donde el primer grupo parece haber captado el dominio a modelar (MacNamara, 2004). siendo consistente con lo expresado por Allen et al. (2015). La alta y baja coherencia semántica se da en los grupos ALTA_N y BAJA_N respectivamente, siendo quienes a su vez concentran los mejores y peores resultados de SUMA_T respectivamente, agrupamiento de datos que no es al azar. Lo antedicho nos estaría permitiendo afirmar que en general se han construido espacios semánticos coherentes (Valle-Lisboa y Mizraji, 2007) en los grupos de estudiantes que tienen mejores calificaciones que entre quienes tienen calificaciones más bajas. Si bien los estudiantes con mejores resultados en SUMA_T son quienes han tenido mejores respuestas en las preguntas, cabría preguntarse si esto se ve reflejado también en la relación entre los conceptos fuertes del discurso y qué tan fluido es su trayecto en el grafo de ideas (Valle-Lisboa et al., 2014); aspecto no estudiado en el presente trabajo.

Las respuestas a conceptos teóricos básicos agrupadas en SUMA_R, muestra que más del 50% de los estudiantes que tienen malos resultados en la construcción del programa (variable BAJA_N) tienen también pobres respuestas a conceptos teóricos. Al mismo tiempo la correlación entre SUMA_R y la NOTA es baja para todas las categorías, como es presentado en los anexos. El test X^2 para estos datos muestra que no existe relación entre los resultados obtenidos en las respuestas teóricas y la calificación final. En vista de estos resultados podríamos estar diciendo que las evaluaciones teóricas, que consisten en explicar conceptos básicos de programación, no serían un buen indicador de cómo los estudiantes pueden resolver la construcción de programas viables (aquellos que tienen las bases conceptuales y procedimentales adecuadas para resolver un programa, en nuestro caso implica mover un vehículo y lograr que el juego presente algunas condiciones de victoria o finalización, pudiendo incluso

operar con variables), aunque sí podrían serlo de la construcción de programas no viables. Cabe preguntarse si las propias preguntas planteadas son adecuadas como predictores de los conocimientos de los estudiante y si las respuestas son de tipo memorísticas o fruto de un análisis y relación de ideas previas. En efecto, podríamos estar frente a un grupo de preguntas que si bien son relevantes desde la programación y el problema planteado, resultan insuficientes como forma de conocer qué saben los estudiantes. Al mismo tiempo deberíamos considerar que en ocasiones los docentes podemos favorecer las respuestas de tipo memorístico (haciendo énfasis en palabras clave o incluso en los propios enunciados) y que los estudiantes pueden entender que aprender implica reproducir literalmente enunciados; aspectos que podrían dar lugar a respuestas que responden parcialmente a la consigna.

El análisis del discurso completo, lematizado o de conceptos fuertes no modifica la tendencia de los resultados, pues a mayor calificación (o grupo de ella) mayores valores de NODOS, ARISTAS y PEA tienen los alumnos. A medida que la cantidad total de palabras disminuye, el coeficiente de clusterización también tiende a cambiar, siendo mayor entre quienes presentan peores resultados; lo cual estaría asociada al uso recurrente de menos términos. La mayor cantidad de palabras en general y términos fuertes en particular, propio de quienes tienen mejores resultados, estaría mostrando que dichas palabras se relacionan más fuertemente con las respuestas esperadas, que hacen a un correcto modelado del problema y que deberían estar en consonancia con las respuestas dadas por los profesores. Los estudiantes que han obtenido peores calificaciones muestran una reducción mayor de conceptos fuertes que el resto, lo que estaría mostrando mayores dificultades para representar el problema. Entendemos que se estaría verificando en los estudiantes que obtienen buenos resultados una superposición semántica con la solución esperada, cabiendo preguntarse si los resultados son análogos a los presentados por Crossley et al. (2019). Debe tener en cuenta que la reducción de palabras evidenciada en el estudio de conceptos fuertes afecta significativamente la cantidad de nodos, lo que podría estar aumentando el ruido y distorsionar los resultados debido a la ausencia de potencia estadística.

Realizado el proceso de clusterización pudimos ver que observamos que dos clusters se han diferenciado claramente del resto: el que agrupa las calificaciones y valores de SUMA_T (variable no usada para la clusterización) más altas, por un lado; y aquellas que agrupan a las calificaciones y valores de SUMA_T más bajo por otro. Todos los alumnos que han obtenido 0 o 1 en SUMA_T, lo que significa tres o más respuestas evaluadas como 0, se encuentran en el mismo cluster. Cuando consideramos la cantidad total de estudiantes vemos que casi dos terceras partes de quienes tienen notas bajas

pertenecen al mismo cluster; la misma relación se observa entre quienes tienen calificaciones altas. Si consideramos la representación que hacen los estudiantes del problema, las mismas tienen relación con los productos efectivamente construidos; por lo que se podría afirmar que sería de esperar que un estudiante que haya presentado un modelo adecuado tenga asociada una calificación alta, e inversamente, si el modelo no lo es, la calificación sería baja. Por último debemos recordar que el cluster en el cual se encuentran estudiantes con calificaciones altas tiene también un valor promedio de SUMA_T alto; inversamente, los cluster que concentran a quienes obtienen peores calificaciones tienen también menores valores promedio de SUMA_T. Resulta interesante que usando la herramienta indicada, hemos verificado resultados similares a agrupar estudiantes por grupo de calificación como también hicimos. En efecto, definir un criterio de agrupamiento basado en la nota permitió ver que la mayoría de los estudiantes que tienen un alto SUMA_T tienen también una calificación, criterio que también se vio cuando no forzamos a un determinado criterio de agrupamiento.

Analizamos también la relación entre las palabras fuertes del discurso, realizando el entrenamiento sobre tres grupos de palabras en función de la calificación de los estudiantes. Se observa que el grupo de palabras dadas forman vectores coherentes, relacionadas semánticamente; desde la programación de la solución, al menos para los estudiantes que han obtenido mejores calificaciones. Dada una palabra que forma parte de la respuesta esperada (como podría ser el caso de Punto), se observa que para quienes obtienen mejores calificaciones las mismas se relacionan fuertemente con otras de la respuesta sobre cómo resolver el problema; aspecto que en general no se da entre quienes han obtenido calificaciones más bajas. Este último grupo de estudiantes muestra resultados menos coherentes, donde las palabras resultado no parecen tener relación significativa con el dominio del problema como era de esperar (MacNamara, 2004), ni con el resto de las palabras que conforman el vector de relaciones semánticas, dando la impresión que no han podido construir espacios semánticos coherentes (Valle-Lisboa y Mizraji, 2007) como sí lo han logrado quienes han obtenido calificaciones más altas.

Debemos considerar que algunas de las categoría a evaluar, como ser la correspondiente a T_MOVIMIENTOS, podía resultar trivial para los estudiantes (ya sea la explicación o incluso su programación). Esta variable ha tomado valores relativamente altos en la tabulación de datos, en tanto que su no consideración en los cálculos no ha incidido significativamente en relación a su inclusión. Dado que la mayoría de los alumnos no han presentado dificultades para expresar cómo sería la programación de los movimientos, cabe preguntarse

si la inclusión de esta variable ha sido apropiada. La solución trivial a cómo resolver el movimiento de los objetos, o un muy buen dominio de los estudiantes de su realización, podría haber generado muy buenos resultados en esta variable, impidiendo diferenciarla del resto de los conceptos.

El programa solicitado así como el lenguaje pudieron estar influyendo en los resultados obtenidos. La consigna en sí misma tenía un grado de concretitud significativo, por lo que deberíamos considerar si esto no estaría incidiendo en la sub valoración del modelado por parte de los estudiantes. En tal sentido, algunos adolescentes pudieron entender que la mejor forma de explicar cómo resolver el problema es la propia solución, aspecto que fue evaluado como negativo en la tabulación de datos, pero que el alumno puede evaluar como suficiente. Otros alumnos pudieron considerar que parte del problema a explicar resultaba sencillo y por tanto que no era necesaria una explicación que fuese más allá una breve descripción Si bien el extremo anterior es válido para todos los estudiantes, entendemos que es más probable que se haya aplicado a quienes tienen un mejor desempeño en programación, pues aumentaría la probabilidad que consideren triviales algunos contenidos. En tal sentido se podrían estar definiendo tres posibles líneas de trabajo complementarias con el fin de continuar analizando el modelado: pedirle a los alumnos que expliquen todo aquello a resolver; solicitarles explicar cómo resolverían el problema a la vez que se les niega la posibilidad de usar una serie de palabras (Miños, 2016b); o bien construir una consigna más genérica y menos atada a la solución visual de los estudiantes.

Pedir que los alumnos expliquen todo aquello que se debía hacer permitiría tener un diccionario de datos mayor, con más posibilidades de análisis semántico, de cohesión y coherencia del texto. Sin embargo esto podría implicar también que el alumno, en su afán de complacer al profesor, agregue información no necesaria o poco relevante. Impedir que los alumnos usen una serie de palabras evitaría que los alumnos usen los mismos términos que deben definir (como usar la palabra condiciones para definir qué es una condición), obligándolos a un esfuerzo cognitivo que permite reestructurar la respuesta. Como contrapartida debería tenerse en cuenta las posibles dificultades que puedan expresar los alumnos en el uso del lenguaje, los sinónimos y el uso de términos que representen la misma idea de una forma distinta a la pensada originalmente. El trabajo con consignas más amplias evitaría el alto grado de concretización de los programas, construyendo soluciones más genéricas, pudiendo observarse así más claramente si existe o no una trayectoria fluida entre los grandes conceptos del texto (Valle-Lisboa et al., 2014). El esfuerzo de parte del estudiante para

representar la solución podría convertirse en un obstáculo, a la vez permitiría manejar más conceptos y la relación entre ellos.

Capítulo 5

Conclusiones

Los estudiantes que han modelado (explicado) mejor cómo sería la solución a realizar, expresaron también mejor cómo es la relación entre las cuatro dimensiones estudiadas. Este grupo de alumnos tiene las mejores calificaciones en la construcción del programa, lo que estaría indicando una posible relación entre las ideas sobre cómo resolver el problema de forma general y las relaciones que deben existir entre sus elementos constitutivos por un lado y la construcción del programa por otro. Al mismo tiempo, los estudiantes que no lograron modelar de forma adecuada la solución, son también aquellos que en general no han podido construir programas viables. Las explicaciones que realizaron los estudiantes, en general son adecuadas desde lo disciplinar para aquellos que obtuvieron calificaciones altas, además de presentar también una redacción coherente, la cual hace referencia a los conceptos fuertes propios del programa.

Agrupados los estudiantes en tres categorías en función de sus calificaciones, hemos verificado una relación directa entre las mismas y la cantidad de palabras, las relaciones que se establecen entre dichas palabras (aristas del grafo) y el porcentaje de exceso de aristas. Los estudiantes con mejores calificaciones presentan siempre relaciones semánticas más complejas, sea por la cantidad de palabras totales, de conceptos fuertes usados o por la relaciones entre distintas palabras y con otros conceptos fuertes desde lo disciplinar. Lo antedicho unido al hecho que mayores valores de SUMA_T se asocian a mejores calificaciones, nos estaría permitiendo afirmar que los estudiantes con mejores resultados

presentan relaciones léxicas más complejas (Skalicky et al., 2017), ya sea por las características de los nodos que lo representan, como por el contenido de las expresiones (variables T en general). Si bien no hemos establecido criterios de cohesión del texto en el sentido dado por Allen et al. (2015), es cierto que la evaluación de las respuestas implica considerar aspectos asociados a ella, estando por tanto en condiciones de afirmar que mayores valores de SUMA_T serían también un indicador de la coherencia de la respuesta. De este modo afirmamos que existen espacios semánticos claramente definidos en función de las calificaciones obtenidas en el sentido dado por Valle-Lisboa y Mizraji (2007).

Es consistente en nuestro trabajo el hecho que independientemente del criterio de análisis y agrupamiento de los resultados, los alumnos que han logrado construir mejores programas (mayores valores de NOTA) son también quienes han tenido mejores registros en la variable SUMA_T. Cabe preguntarse si la relación que se evidencia entre la construcción de buenos resúmenes y el conocimiento del texto original descrito por Crossley et al. (2019) no se estaría aplicando también en este caso, de modo que quienes representan mejor cómo resolver el problema son también quienes lo han comprendido mejor; pregunta que en principio entendemos debería ser respondida afirmativamente Dado que no hemos podido demostrar causalidad, entendemos que sería una hipótesis a considerar en futuros trabajos.

La variable SUMA_T además de representar el discurso como un todo, da cuenta de lo que entendemos son en este caso cuatro dimensiones disciplinares fundamentales a la hora de modelar el problema propuesto. En tal sentido, los alumnos con registros más altos de SUMA_T son también quienes están logrando identificar los grandes elementos constitutivos del problema en el sentido dado por Soloway (1986) y Ebrahimi (1994). En vista de la relación entre SUMA_T y las calificaciones obtenidas, podríamos estar afirmando que se está verificando lo expresado por Winslow (1996) pues quienes logran comprender y modelar la semántica del problema a resolver, son quienes también logran hacer uso de la sintaxis adecuada para la solución del problema. El grupo de estudiantes que obtuvo puntuaciones bajas en las variables involucradas en SUMA_T, y por tanto en esta variable, presentó como errores recurrentes la ausencia de respuestas, la no correctitud, vaguedad o el uso de ejemplos sin referencia clara al concepto a trabajar. Es así que se observa que los resultados obtenidos por los estudiantes del grupo medio bajo estarían en consonancia con lo planteado por Meerbaum-Salant et al. (2013), pues existe una tendencia a la simplificación de los conceptos a usar, expresado esto en el alto grado de concretización del saber.

Reiterando el hecho que no hemos podido establecer causalidad, entendemos que algunos indicadores estarían sugiriendo un buen desempeño de los estudiantes en la construcción de este tipo de programas. Aspectos como la cantidad de palabras usadas y la relación entre estas y otras, estarían relacionadas con buenas explicaciones de las soluciones a modelar. Al mismo tiempo las buenas explicaciones se relacionan con la coherencia en la redacción (pues son determinantes para construir enunciados correctos), así como la mención y el desarrollo de las ideas principales del problema a resolver desde lo disciplinar (relacionado también a explicaciones adecuadas). La relación entre los conceptos fuertes del discurso (que hacen a las características de la solución) sería un indicador de coherencia y cohesión de la solución, la cual a su vez tiene relación con la solución a construir por parte del estudiante.

Como síntesis general podemos observar que aquellos estudiantes que representan mejor el problema, son también quienes logran resolverlo de mejor forma. Es así que desde una perspectiva didáctica, y considerando el hecho que la explicación que los alumnos hacen de cómo se debe resolver el problema sería un predictor del éxito en la construcción de los programas, deberíamos hacer énfasis en este aspecto en lugar de las preguntas asociadas a elementos teóricos o el funcionamiento de sentencias del lenguaje. Este enfoque de análisis debería tener en cuenta las ideas más importantes del problema a resolver, así como la coherencia en la redacción, aspectos que deberían ser tratados en clase conjuntamente con los alumnos.

Bibliografía

Aho, A. V. (2012). Computation and Computational Thinking. *The Computer Journal*, 55(7), 832–835. doi:10.1093/comjnl/bxs074

Aho, A., Hopcroft, J. y Ullman, J. (1998). *Estructuras de datos y algoritmos* . México: Adisson Wesley Longman.

Allen, L. K., Snow, E. L., y McNamara, D. S. (2015). Are you reading my mind?: modeling students' reading comprehension skills with natural language processing techniques. En *Proceedings of the Fifth International Conference on Learning Analytics And Knowledge - LAK '15*. doi:10.1145/2723576.2723617

Altszyler, E., Sigman, M., Ribeiro, S., y Slezak, D. F. (2016). Comparative study of LSA vs Word2vec embeddings in small corpora: a case study in dreams database. *arXiv preprint arXiv:1610.01520*.

Antúnez, S. del Carmen, L., Imbernon, F., Parcerisa, S. y Zabala, A. (1992). *Del proyecto educativo a la programación de aula*. España: Editorial GRAÓ.

Armoni, M., Meerbaum-Salant, O., y Ben-Ari, M. (2015). From Scratch to "Real" Programming. *ACM Transactions on Computing Education*, 14(4), 1–15. doi:10.1145/2677087

Arsac, J. (1988). La didactique de l'informatique: un problème ouvert?. En *Colloque francophone sur la didactique de l'informatique.* Association EPI, (pp. 9-18).

Astolfi, J. (1999). *El error, un medio para enseñar*. Sevilla: Diada Editora.

Blackwell, A. F. (2002, June). What is programming?. En *14 th Workshop of the Psychology of Programming Interest Group*, Brunel University, pp. 204 - 218.

Brennan, K., y Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. En *Proceedings of the 2012 annual meeting of the American educational research association*, Vancouver, Canada (Vol. 1, p. 25).

Bouvier, D., Lovellette, E., Matta, J., Alshaigy, B., Becker, B. A., Craig, M., Jackova, J., McCartney, R., Sanders, K. y Zarb, M. (2016). Novice programmers and the problem description effect. En *Proceedings of the 2016 ITiCSE Working Group Reports* (pp. 103-118).

Cabana, Á., Valle-Lisboa, J. C., Elvevåg, B., y Mizraji, E. (2011). Detecting order–disorder transitions in discourse: Implications for schizophrenia. *Schizophrenia research*, *131*(1-3), 157-164.

Carroll, J. y Sapon, S. (1959). Modern Language Aptitude Test. San Antonio, TX: Psychological Corporation.

Chomsky, W. (1957). *Hebrew: The eternal language*. Philadelphia: Jewish Publication Society

Clements, D. H., y Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, 76(6), 1051–1058. doi:10.1037/0022-0663.76.6.1051

Cunningham, K., Blanchard, S., Ericson, B., y Guzdial, M. (2017). Using Tracing and Sketching to Solve Programming Problems. En *ACM Conference on International Computing Education Research - ICER '17*. doi:10.1145/3105726.3106190

Compañ-Rosique, P., Satorre-Cuerda, R., Llorens-Largo, F. y Molina-Carmona, R. (2015). Enseñando a programar: un camino directo para desarrollar el pensamiento computacional. RED. *Revista de Educación a Distancia*. Número 46.

Congacha, G. y Valladolid, C. (s.f.). Análisis no supervisado: K - Means. Recuperado de https://giorgiacongacha.shinyapps.io/kmeans/

Crossley, S. A., Kim, M., Allen, L., y McNamara, D. (2019). Automated Summarization Evaluation (ASE) Using Natural Language Processing Tools. *Artificial Intelligence in Education*, 84–95. doi:10.1007/978-3-030-23204-7_

Davies, S. P. (1993). Models and theories of programming strategy. *International Journal of Man-Machine Studies*, 39(2), 237–267. doi:10.1006/imms.1993.1061

Denning, P. J. (1989). A debate on teaching computing science. *Communications of the ACM*, 32(12), 1397–1414. doi:10.1145/76380.76381

Denning, P., Comer, D., Gries, D., Mulder, M., Tucker, A., Turner, A. y Young, P. (1989). Computing as a discipline. *Communications of the ACM*, 32(1), 9 – 23.

Denning, P. J. (2005). Is computer science science? *Communications of the ACM*, 48(4), 27. doi:10.1145/1053291.1053309

Dirección General de Enseñanza Secundaria (2016). Planes de asignaturas. Recuperado de https://www.ces.edu.uy/index.php/propuesta-educativa/20234

Dirección General de Educación Técnico Profesional (s.f.). PROGRAMAS | Programas planeamiento educativo. Recuperado de https://planeamientoeducativo.utu.edu.uy/diseno-y-desarrollo-curricular/programas

Dodig-Crnkovic, G. (2002, April). Scientific methods in computer science. En *Conference for the Promotion of Research in IT* at New Universities and at University Colleges in Sweden, Skövde, Suecia (pp. 126-130).

Dowek, G. (2005). Quelle informatique enseigner au lycée. *Bulletin de l'APMEP*, (480).

Ebrahimi, A. (1994). Novice programmer errors: Language constructs and plan composition. I*nternational Journal of Human-Computer Studies*, 41(4), 457-480.

Ehlert, A., & Schulte, C. (2009). Empirical comparison of objects-first and objects-later. Proceedings of the *Fifth International Workshop on Computing Education Research Workshop - ICER '09*. doi:10.1145/1584322.1584326

Fedorenko, E., Ivanova, A., Dhamala, R., y Bers, M. U. (2019). The Language of Programming: A Cognitive Perspective. *Trends in Cognitive Sciences*. doi:10.1016/j.tics.2019.04.010

Fernández, L., Peña, R., Nava, F. y Velázquez, A. (2002). Análisis de las propuestas de la enseñanza de la programación orientada a objetos en los primeros cursos. En *Actas de las VIII Jornadas de Enseñanza Universitaria de la Informática (JENUI'02)*. Cáceres, España: Universidad de Extremadura, 433-440.

Fiore, E., y Leymonié, J. (2007). *Didáctica práctica para Enseñanza Media y Superior*. Montevideo: Editorial Grupo Magro.

Fitzgerald, S., Simon, B., y Thomas, L. (2005, October). Strategies that students use to trace code: an analysis based in grounded theory. En Proceedings of the first international workshop on Computing education research (pp. 69-80).

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., y Resnick, M. (2013). Designing ScratchJr. En *Proceedings of the 12th International Conference on Interaction Design and Children - IDC '13*. doi:10.1145/2485760.2485785

Frawley, W. (2013). *Linguistic semantics*. Routledge.

Gal'perin, P. (1989). Mental Actions as a Basis for the Formation of Thoughts and Images. *Soviet Psychology*, 27(3), 45–64. doi:10.2753/rpo1061-0405270345

Grogono, P. (1984). Programación en Pascal. Washington: ADDISON - WESLEY IBEROAMERICANA S.A.

Gutiérrez, C. M. (2006). Semántica cognitiva: modelos cognitivos y espacios mentales. *A Parte Rei: revista de filosofía*, *43*(5).

Grandell, L., Peltomäki, M., Back, R. J., y Salakoski, T. (2006, January). Why complicate things? Introducing programming in high school using Python. En *Proceedings of the 8th Australasian Conference on Computing Education*, V. 52 (pp. 71-80).

Grover, S., y Basu, S. (2017, March). Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic. En *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 267-272).

González, L. (2005). Espacios mentales. *Contrastes. Revista Internacional de Filosofía*.

Guo, P. J. (2018). Non-Native English Speakers Learning Computer Programming. En *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. doi:10.1145/3173574.3173970

Harris, A. (1954). *Distributional Structure, WORD*, 10:2-3, 146-162, DOI:10.1080/00437956.1954.11659520

Hernández, R., Fernández, C. y Baptista, P. (1991). *Metodología de la Investigación*. (5.ª ed.). México: McGraw-Hill.

Herrera, J. A. V. (2016). ¿ Por qué el estudio del lenguaje es fundamental para la cognición?. *Sophia: Colección de Filosofía de la Educación*, (20), 67-85.

Hinojosa Rivero, G. (2008). El tratamiento estadístico de las redes semánticas naturales. *Revista Internacional de Ciencias Sociales y Humanidades, SOCIOTAM*, XVIII(1),133-154.

Hoc, T.R.G. Green, R. Samurçay, y D.J. Gillmore (Eds.), *Psychology of programming* (pp. 157–174). Londres: Academic Press.

Johnsonbaugh, R. (2005). *Matemáticas discretas*. México: Pearson Educación.

Johnson-Laird, P. N. (2010). Mental models and human reasoning. *Proceedings of the National Academy of Sciences*, 107(43), 18243-18250.

Kaasbøll, J. J. (1998). Exploring didactic models for programming. En *NIK 98 – Norwegian Computer Science Conference* (pp. 195-203).

Kaučič, B., y Asič, T. (2011, May). Improving introductory programming with Scratch?. En *2011 Proceedings of the 34th International Convention MIPRO* (pp. 1095-1100).

Kintsch, W. (1998) The Representation of Knowledge in Minds and Machines, International Journal of Psychology, 33:6, 411-420, DOI: 10.1080/002075998400169

Kintsch, W., y van Dijk, T. A. (1978). Toward a model of text comprehension and production. *Psychological Review, 85*(5), 363–394. https://doi.org/10.1037/0033-295X.85.5.363

Kordaki, M. (2012). Diverse categories of programming learning activities could be performed within Scratch. *Procedia-Social and Behavioral Sciences*, 46, 1162-1166.

Koulouri, T., Lauria, S., y Macredie, R. D. (2014). Teaching Introductory Programming. *ACM Transactions on Computing Education*, 14(4), 1–28. doi:10.1145/2662412

Kruchten, P. B. (1995). The 4+ 1 view model of architecture. *IEEE software*, *12*(6), (pp. 42 - 50).

Lahtinen, E., Ala-Mutka, K., y Järvinen, H. M. (2005). A study of the difficulties of novice programmers. *Acm sigcse bulletin*, 37(3), 14-18.

Landauer, T. K., y Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2), 211.

Liao, Y., y Bright, G. (1991). Effects of computer programming on cognitive outcomes: A meta-analysis. J*ournal of educational computing research*, 7(3), 251-268. doi:10.2190/e53g-hh8k-ajrr-k69m

Litwin, E. (1997). *Las Configuraciones Didácticas*. Buenos Aires: Paidós.

Lye, S., y Koh, J. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. doi:10.1016/j.chb.2014.09.012

Ma, L., Ferguson, J., Roper, M., y Wood, M. (2011). Investigating and improving the models of programming concepts held by novice programmers. *Computer Science Education*, 21(1), 57–80. doi:10.1080/08993408.2011.55472

McNamara, D. S. (2004). SERT: Self-Explanation Reading Training. *Discourse Processes*, 38(1), 1–30. doi:10.1207/s15326950dp3801_1

MacQueen, J. (1967, June). Some methods for classification and analysis of multivariate observations. En *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281-297).

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., y Rusk, N. (2008, March). Programming by choice: urban youth learning programming with scratch. En *Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 367-371).

Maloney, J., Resnick, M., Rusk, N., Silverman, B., y Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, 10(4), 16.

Meerbaum-Salant, O., Armoni, M., y Ben-Ari, M. (2011). Habits of programming in scratch. En *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education - ITiCSE '11.* doi:10.1145/1999747.1999796

Meerbaum-Salant, O., Armoni, M., y Ben-Ari, M. (2013). Learning computer science concepts with scratch. *Computer Science Education*, 23(3), 239-264.

Mikolov, T., Chen, K., Corrado, G., y Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv* preprint arXiv:1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., y Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems,* 3111-3119.

Mikolov, T., Yih, W. T., y Zweig, G. (2013, June). Linguistic regularities in continuous space word representations. En *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies.* (pp. 746-751).

Miños, A. (2016a). Uso didáctico de estrategias inductivas en un curso introductorio de programación estructurada. *Tecné, episteme y didax*i, (39), 95-110. doi.org/10.17227/01203916.4583

Miños, A. (2016b). Del decímelo con tus palabras al no usar las palabras... Una experiencia de evaluación desde la didáctica de la informática. *InterCambios*, 3(2), 77 - 81.

Miños, A. (2017). Elementos estructurantes de la Didáctica de la Informática. *Virtualidad, Educación y Ciencia*, 8(14), 100-110.

Mota, N.B (2017). *Mapeamento mental através da análise computacional do discurso* (Tesis de doctorado, Universidade Federal do Rio Grande do Norte). Recuperado de https://repositorio.ufrn.br/handle/123456789/24681.

Mota, N., Furtado, R., Maia, P., Copelli, M. y Ribeiro, S. (2014) Graph analysis of dream reports is especially informative about psychosis. *Scientific reports*, 4(3691). DOI:10.1038/srep03691.

Mota, N.B., Sigman, M., Cecchi, G. et al. (2018) The maturation of speech structure in psychosis is resistant to formal education. *NPJ Schizophr,* 4(25). https://doi.org/10.1038/s41537-018-0067-3

Mota, N. B., Weissheimer, J., Madruga, B., Adamy, N., Bunge, S. A., Copelli, M., y Ribeiro, S. (2016). A Naturalistic Assessment of the Organization of Children's Memories Predicts Cognitive Functioning and Reading Ability. *Mind, Brain, and Education*, 10(3), 184–195. doi:10.1111/mbe.12122

Muñoz, R., Barcelos, T. S., Villarroel, R., Barría, M., Becerra, C., Noel, R., y Frango Silveira, I. (2015, July). Uso de Scratch y Lego Mindstorms como apoyo a la docencia en Fundamentos de programación. En *Actas de las XXI Jornadas de*

la Enseñanza Universitaria de la Informática (pp. 248-254). Universitat Oberta La Salle.

Pascual, E. (2012). Los espacios mentales y la integración conceptual. *Lingüística cognitiva. Barcelona: Anthropos*, pp. 147 - 166

Pesce, F. (2008). La especificidad de la didáctica en la formación docente: mirada interpretativa y proyectiva. *Voces*, Año X(27), 9– 2.

Pham, D. T., Dimov, S. S., y Nguyen, C. D. (2005). Selection of K in K-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1), 103–119. doi:10.1243/095440605x8298

Prat, C. S., Madhyastha, T. M., Mottarella, M. J., y Kuo, C. H. (2020). Relating natural language aptitude to individual differences in learning programming languages. *Scientific reports*, 10(1), 1-10.

Pressman, R., (2010). *Ingeniería de software. Un enfoque práctico* (7ª ed.). México: Mc Graw Hill Educación.

Rabazo, M., Moreno, J. y Rabazo, A. (2008). Aportaciones de la psicología de Vygotsky a la enseñanza de la producción textual. *International Journal of Developmental and Educational Psychology*, (4), 473-482.

Řehůřek, R. (2021). Word2Vec. Recuperado de https://radimrehurek.com/gensim/models/word2vec.html

Renumol, V. G., Janakiram, D., y Jayaprakash, S. (2010). Identification of Cognitive Processes of Effective and Ineffective Students During Computer Programming. *ACM Transactions on Computing Education*, 10(3), 1–21. doi:10.1145/1821996.1821998

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B. y Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, 52(11), 60-67.

Robins, A., Rountree, J., y Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 13(2), 137–172. doi:10.1076/csed.13.2.137.14200

Rogalski, J., y Samurçay, R. (1990). Acquisition of programming knowledge and skills. En J.M. Hoc, T.R.G. Green, R. Samurçay,y D.J. Gillmore (Eds.), Psychology of programming (pp. 157–174). London: Academic Press.

Sanders, I., Galpin, V., y Götschi, T. (2006). Mental models of recursion revisited. *ACM SIGCSE Bulletin,* 38(3), 138 - 142.

Scaffidi, C., y Chambers, C. (2012). Skill progression demonstrated by users in the Scratch animation environment. *International Journal of Human-Computer Interaction*, 28(6), 383-398.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. Educational Research Review, 22, 142–158. doi:10.1016/j.edurev.2017.09.003

Simabucuru, G., Copelli, M., Ribeiro, S., y Mota, N. (2020). S199. How to collect free speech to speech graph analysis: standardized time-limited protocol. *Schizophrenia Bulletin,* 46 (1), S114. https://doi.org/10.1093/schbul/sbaa031.265

Skalicky, S., Crossley, S., McNamara, D. y Muldner, K. (2017) Identifying Creativity During Problem Solving Using Linguistic Features. *Creativity Research Journal*, 29(4), 343-353, doi:10.1080/10400419.2017.1376490

Soloway, E. (1986). Learning to program= learning to construct mechanisms and explanations. *Communications of the ACM*, 29(9), 850-858.

Tucker, A. (2003). A model curriculum for k--12 computer science: Final report of the acm k--12 task force curriculum committee. *ACM*. doi.org/10.1145/2593247

Valle-Lisboa, J. C., y Mizraji, E. (2007). The uncovering of hidden structures by Latent Semantic *Analysis. Information Sciences*, 177(19), 4122–4147. doi:10.1016/j.ins.2007.04.007

Valle-Lisboa, J. C., Pomi, A., Cabana, Á., Elvevåg, B., y Mizraji, E. (2014). A modular approach to language production: Models and facts. *Cortex*, 55, 61–76. doi:10.1016/j.cortex.2013.02.005

Van Dijk, T. (2004). Discurso y dominación. *Grandes conferencias en la Facultad de Ciencias Humanas*, *4*, pp. 5 - 28.

Van Dijk, T. (2005). *Estructuras y funciones del discurso: una introducción interdisciplinaria a la lingüística del texto ya los estudios del discurso*. Siglo XXI.

Van Dijk, T. (2016). *Discurso y conocimiento: Una aproximación sociocognitiva* (Vol. 302632). Editorial Gedisa.

Van Dijk, T. (2019). *El discurso como interacción social*. Editorial Gedisa.

van Emden, M. H. (2015). *Elements of Programming*. Andromeda Research Associates, Ltd

Vázquez-Cano, E., y Delgado, D. F. (2015). La creación de videojuegos con Scratch en Educación Secundaria. *Communication papers*, 4(06), 63 - 73.

Villalobos, J. A., y Calderón, N. A. (2009). Proyecto Cupi2: un enfoque multidimensional frente al problema de enseñar y aprender a programar. *Revista De Investigaciones UNAD*, 8(2), 45-64. https://doi.org/10.22490/25391887.635

Watts, D. J., y Strogatz, S. H. (1998). Collective dynamics of “small-world” networks. *Nature*, 393(6684), 440–442. doi:10.1038/30918

Webb, M., Davis, N., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., y Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when?. *Education and Information Technologies*, 22(2), 445-468.

Wiedenbeck, S., Ramalingam, V., Sarasamma, S., y Corritore, C. (1999). A comparison of the comprehension of object-oriented and procedural programs by novice programmers. *Interacting with Computers*, 11(3), 255–282. doi:10.1016/s0953-5438(98)0002

Wiedenbeck, S., y Ramalingam, V. (1999). Novice comprehension of small programs written in the procedural and object-oriented styles. *International Journal of Human-Computer Studies*, 51(1), 71–87. doi:10.1006/ijhc.1999.0269

Willing, P., Astudillo, G. J., y Bast, S. G. (2010). Aprender a programar (¿y a pensar?) jugando. En *V Congreso de Tecnología en Educación y Educación en Tecnología*.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.

Wing, J. M. (2010). Computational Thinking: What and Why? Recuperado de http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Winslow, L.E. (1996). Programming pedagogy – A psychological overview. *SIGCSE Bulletin, 28*, 17 –22.

Wolz, U., Leitner, H. H., Malan, D. J., y Maloney, J. (2009). Starting with scratch in CS 1. En *Proceedings of the 40th ACM Technical Symposium on Computer Science Education - SIGCSE '09*. doi:10.1145/1508865.1508869

Xie, B, Loksa, D., Nelson, G. Davidson, M., Dong, D., Kwik, H., Hui Tan, A., Hwa, L., Li, m. y Ko, A. (2019) A theory of instruction for introductory programming skills. *Computer Science Education*, 29:2-3, 205-253, DOI: 10.1080/08993408.2019.1565235

Xie, B., Nelson, G. L., y Ko, A. J. (2018). An Explicit Strategy to Scaffold Novice Program Tracing. En *Proceedings of the 49th ACM Technical Symposium on Computer Science Education - SIGCSE '18*. doi:10.1145/3159450.3159527

Anexos

Explicaciones de los profesores

Reviso si el auto está en el circuito. Pueden ocurrir dos cosas 1) que esté en el circuito o 2) que no esté en el circuito 1) Si el auto está en el circuito reviso si estoy en una marca. Si estoy en una marca sumo 10 puntos y avanzo 2 pasos. Si no estoy en una marca avanzo 2 pasos. Al avanzar revisamos si a) el auto está afuera del circuito en ese caso el auto explota y termina el juego o b) si el auto cruzó la línea, en ese caso se muestran los puntos, gana y termina el juego. Vuelvo nuevamente al inicio y reviso si el auto está en el circuito.2) Si no está en el circuito estalla el auto y termina el juego.

Explicar que todo lo que el juego permita hacer y todas las “reacciones” del juego a nuestras acciones las debemos programar. Los elementos u objetos que el programa contiene los debemos diseñar, es decir hacer la pista, los autos, etc.. Los movimientos de los autos se lo debemos asignar a determinadas teclas, eso lo debemos programar. También debemos decidir que cantidad se mueve el auto cuando queremos avanzar o retroceder. Cada vez que un auto toca una marca se logran 10 puntos, esa información se debe registrar en la memoria, de forma que el programa pueda ir sumando los puntos obtenidos. Al salir de la pista el auto explota, eso lo podemos programar en base a los colores, o sea, siempre que el auto toque el color que está fuera de la pista, que explote y el juego termine. También en base a los colores se puede programar que cuando un auto toca el color de la línea de meta el juego termine y muestre los puntos que tenía el auto, este dato debe estar en la memoria.

Bueno, primero deberían fijar el auto en cierta dirección y posición. Luego tendría que tener algunas estructuras – repetir que si toca el color verde desaparezca – o sea – programar varias partes. Luego – repetir que si toca el color blanco gana y ahí mostrar el puntaje. Por ende también deberían crear una variable –

puntos=0 – si toca una marca que incremente en 10 – puntos=puntos+10. Y por ahí sería, creo que no me olvido de nada.

En primer lugar programo el movimiento del auto para que pueda al presionar una flecha de dirección, apuntar a esa dirección y moverse 2 pasos. Luego, el sensor de checkpoint que cuando toque al auto sume 10 puntos a una variable previamente creada. Por otro lado, programo el sensor que detecte cuando toque la zona fuera de la pista el programa se detenga y por último el sensor que al tocar la línea muestre el puntaje obtenido. Las preguntas de los sensores deben realizarse todo el tiempo de ejecución del programa mediante un bucle repetitivo por siempre.

Es decir, una posible solución es que si hay verde el auto trate de volver al gris, si hay gris avance. Si los estudiantes ya tuvieron una base de programación y tienen más conocimientos, es bueno el ejercicio de resolverlo en lenguaje natural sin instrucciones dadas por el docente y luego pasarlo a un lenguaje especifico determinado por el mismo. Bueno en esa solución habría que tener en cuenta los conos también, podría ser, Si detecta verde o anaranjado, el auto gire hasta detectar gris. Si no, el auto avanza. Limitando específicamente a las teclas, podría ser algo así como posible solución : Repetidamente hacer Si detecta verde o anaranjado, Mientras no sea gris presione ➡□ Presione ⬇□ ⬆□

para lograr jugar a eso se deberían programar algunas acciones que permitan recorrer la pista como que el auto avance y gire hacia un lado u otro en función de que la pista tiene curvas y programar algunos hechos como pueden ser salirse de la pista, detectar esas referencias que hay en el pista para sumar puntos y detectar además cuándo se llega a la línea de final y por otro lado crear además algún contador de puntos.

```
puntos = 0
Mientras(!tocaLineaFin y !tocaVerde)
        avanzar(2 pasos)
        ofrecer: doblar(derecha 15°) o doblar(izq 15°) o hacerNada()
        si (tocaMarca)
                puntos = puntos + 10


si (tocaVerde)
         destruir y mostrar Perdió

si(tocaLineaFin)
        mostrar Ganó!
        mostrar puntos
```

Otros resultados relevantes obtenidos con K-Means

SUMA_T de 4 Variables					
	C_1	C_2	C_3	C_4	C_5
ALTO_N	4	13	4	6	3
MEDIO_N	2	9	5	3	12
BAJO_N	2	10	10	2	10

Correlaciones entre la variable Nota y las indicadas agrupadas por calificaciones

ALTA_N					
	SUMA_T				
	Pearson	0,1012			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,1012		corr	0,1012
	std err	0,1880		std err	0,1857
	p-value	0,5947		p-value	0,5978
	SUMA_R				
	Pearson	0,1463			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,1463		corr	0,1463
	std err	0,1869		std err	0,1857

	p-value	0,4404		p-value	0,4438
	PEA				
	Pearson	-0,0655			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	-0,0655		corr	-0,0655
	std err	0,1886		std err	0,1857
	p-value	0,7309		p-value	0,7331
	SUMA_T				
	Pearson	0,4264			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,4264		corr	0,4264

MEDIA_N	std err	0,1680		std err	0,1826
	p-value	0,0168		p-value	0,0159
	SUMA_R				
	Pearson	0,1210			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,1210		corr	0,1210
	std err	0,1843		std err	0,1826
	p-value	0,5168		p-value	0,5200
	PEA				
	Pearson	0,1289			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,1289		corr	0,1289

	std err	0,1841		std err	0,1826
	p-value	0,4895		p-value	0,4927
BAJA_N					
	SUMA_T				
	Pearson	0,2473			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,2473		corr	0,2473
	std err	0,1713		std err	0,1741
	p-value	0,1586		p-value	0,1598
	SUMA_R				
	Pearson	-0,0130			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	

	corr	-0,0130		corr	-0,0130
	std err	0,1768		std err	0,1741
	p-value	0,9417		p-value	0,9422
	PEA				
	Pearson	0,2140			
	Pearson's coeff (t test)			Pearson's coeff (Fisher)	
	corr	0,2140		corr	0,2140
	std err	0,1727		std err	0,1741
	p-value	0,2242		p-value	0,2262

Correlaciones entre la variable Nota y las indicadas para toda la población

NODOS				

Pearson	0,2583			
Pearson's coeff (t test)			Pearson's coeff (Fisher)	
corr	0,2583		corr	0,2583
std err	0,1002		std err	0,1031
p-value	0,0115		p-value	0,0112

ARISTAS				
Pearson	0,2410			
Pearson's coeff (t test)			Pearson's coeff (Fisher)	
corr	0,2410		corr	0,2410
std err	0,1006		std err	0,1031
p-value	0,0187		p-value	0,0184

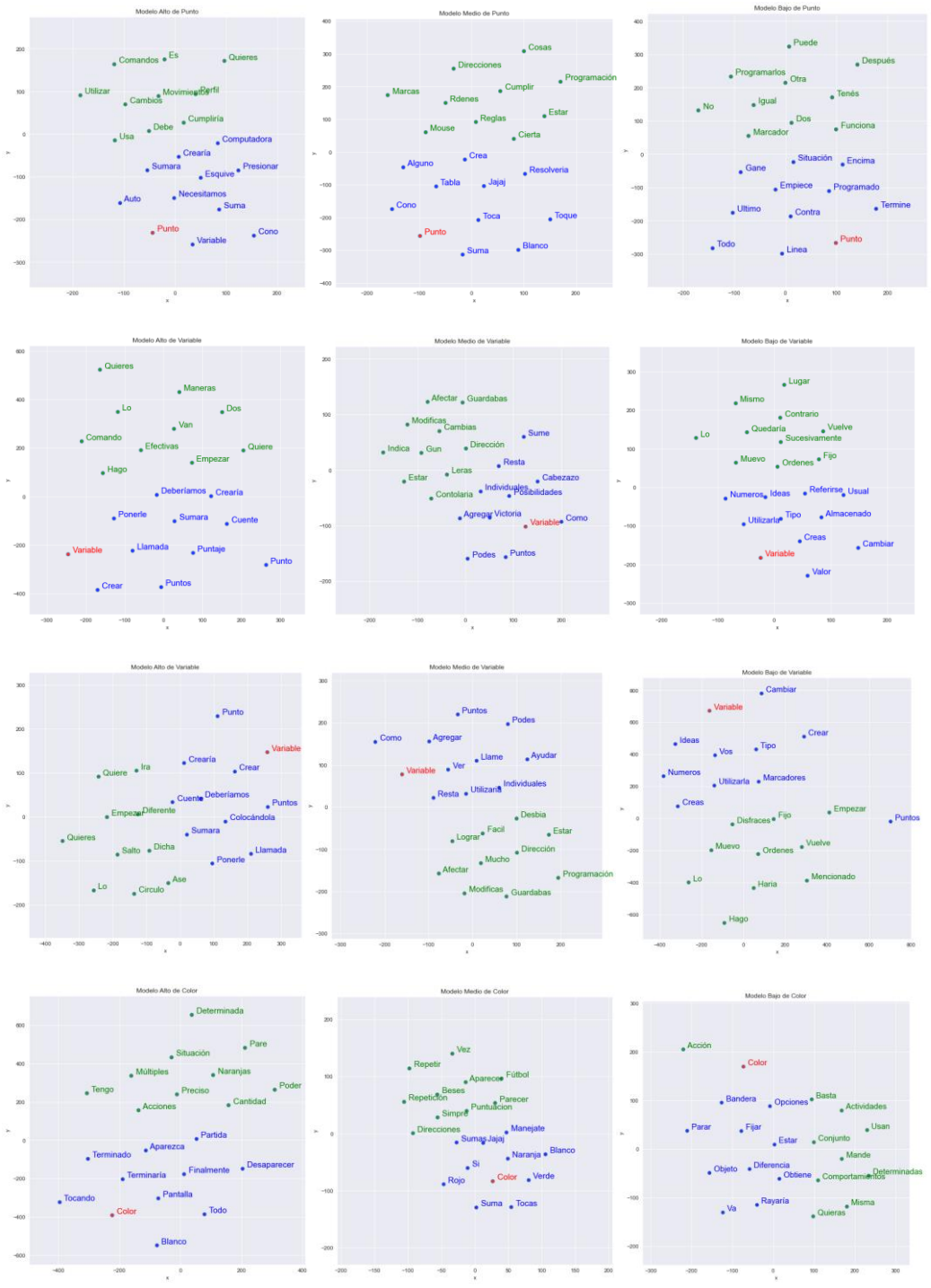
Algoritmo K-Means

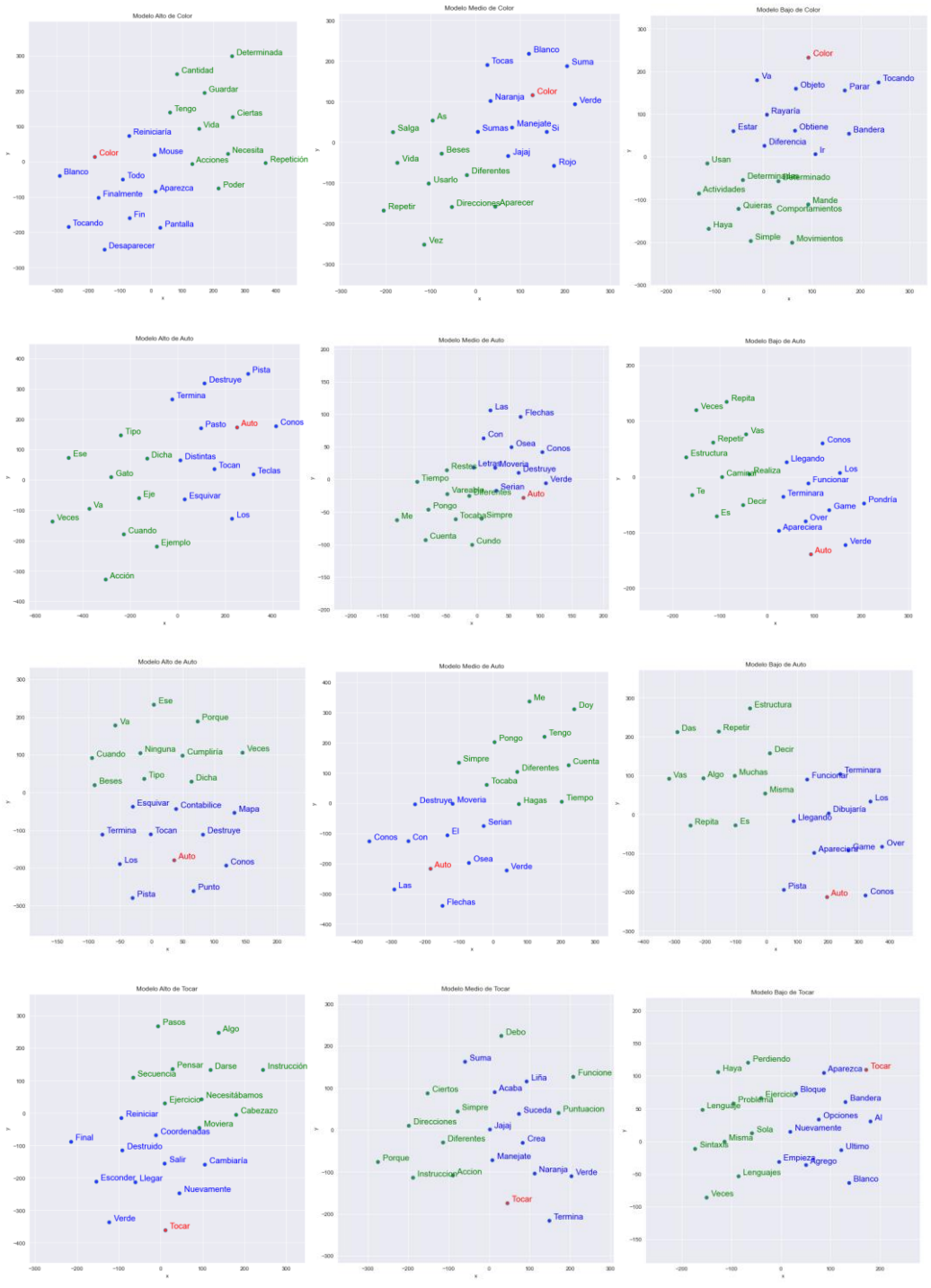
```
server <- function(input, output) {


    # reactivo
  data <- reactive({
    file1 <- input$file
    if(is.null(file1)){return()}
    read.table(file=file1$datapath, sep=input$sep,dec=input$dec, header =
T )

  })

  # Base de datos
  output$table <- renderTable({
    if(is.null(data())){return ()}
    data()
  })

  # coeficiente silueta
  output$coef <- renderTable({
    dfc1<-data.frame(data())
    d <- dist(dfc1)
    avgS <- c() #aqui guardare mis resultados
    for(k in 2:6) {
      cl <- kmeans(dfc1, centers = k, iter.max = 200)
      s <- silhouette(cl$cluster, d)
      avgS <- c(avgS, mean(s[,3]))
    }

    data.frame(nClus = 2:6, Silh = avgS)

  })

  #Garfico cluster
  output$cluster <- renderPlot({
    if(is.null(data())){return ()}

    dfc1<-data.frame(data())
    mc1 <- kmeans(x = dfc1, input$n)
    mc1
    aggregate(dfc1,by = list(mc1$cluster),FUN = mean)
    mydata<- data.frame(dfc1,mc1$cluster)
    points(mc1$centers, col = 1:2, pch = 11)

    fviz_cluster(object = mc1, data = dfc1, show.clust.cent = TRUE,
                 ellipse.type = "euclid", star.plot = TRUE, repel = TRUE)
+
      labs(title = "Resultados clustering K-means") +
      theme_bw() +
      theme(legend.position = "none")


  })
```

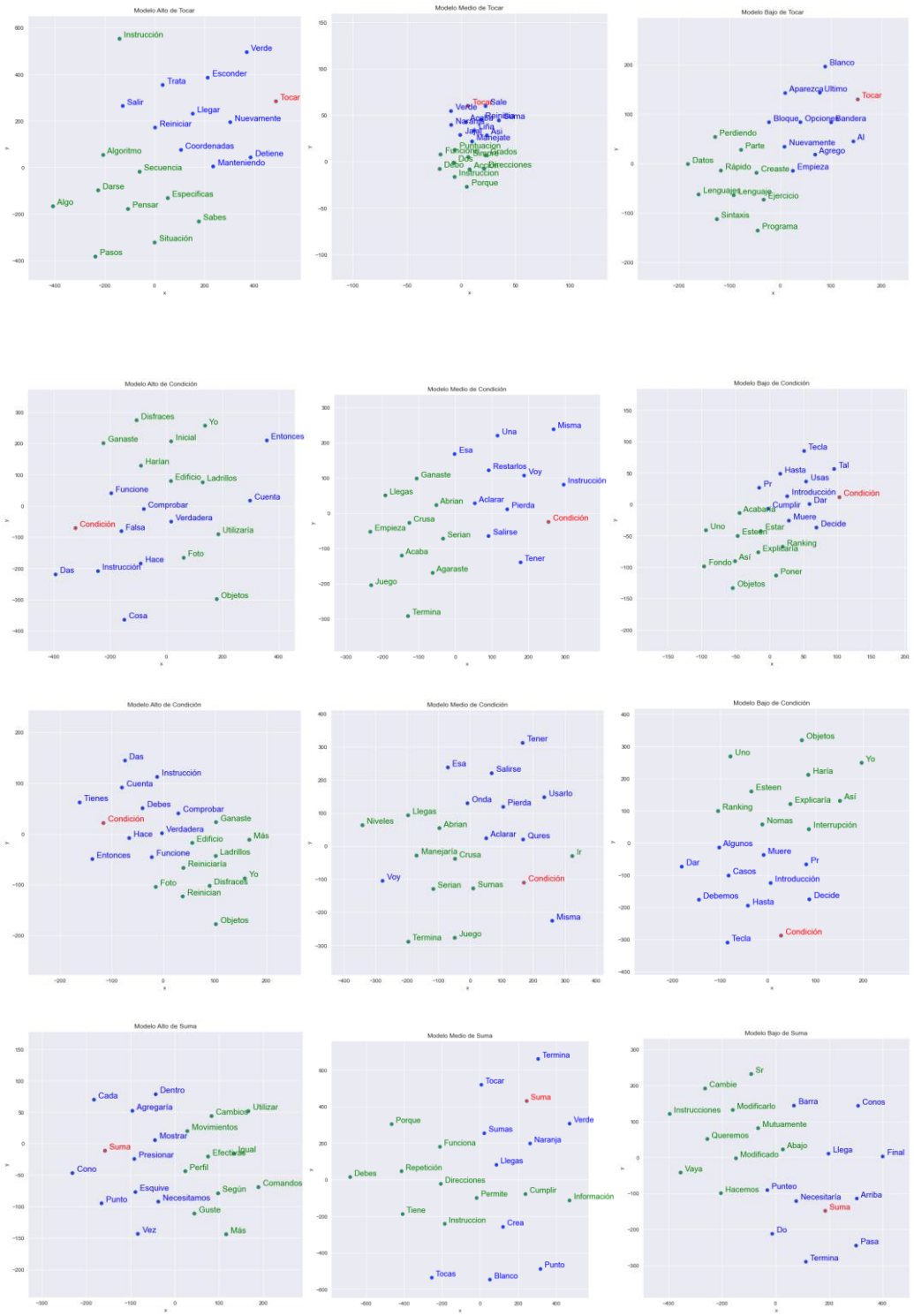
```
    # Asignacion
    output$clasificacion <- renderTable({
      if(is.null(data())){return ()}
      dfc1<-data.frame(data())
      mc1<-kmeans(dfc1,input$n)
      aggregate(dfc1,by = list(mc1$cluster),FUN = mean)
      mydata<- data.frame(dfc1,mc1$cluster)
      mydata

    })

    #Perfiles
    output$perfil <- renderTable({
      dfc1<-data.frame(data())
      mc1<-kmeans(dfc1,input$n)
      mc1$centers
    })

  output$tb <- renderUI({
    if(is.null(data()))
      h5(tags$img(src="cluster.gif", heigth=700, width=700))
    else
      tabsetPanel(
                  tabPanel("Data", tableOutput("table")),
                  tabPanel("C.Silueta",tableOutput("coef")),
                  tabPanel("Cluster", plotOutput("cluster")),
                  tabPanel("Asignacion", tableOutput("clasificacion")),
                  tabPanel("Perfiles", tableOutput("perfil"))
                  )
  })

}



library(shiny)
library(ggplot2)
library(stringr)
library(dplyr)
library(DT)
library(tools)
library(foreign)
library(cluster)
library(Hmisc)
library(factoextra)

ui <- fluidPage(theme = "formato.css",
  titlePanel("ANALISIS NO SUPERVISADO: K - MEANS"),
  sidebarLayout(
    sidebarPanel(
      wellPanel(
      helpText("Su base no debe tener variables cualitativas y debe tener
extension
               .csv o .txt"),
      fileInput("file","Cargue su base de datos")
      ),
```

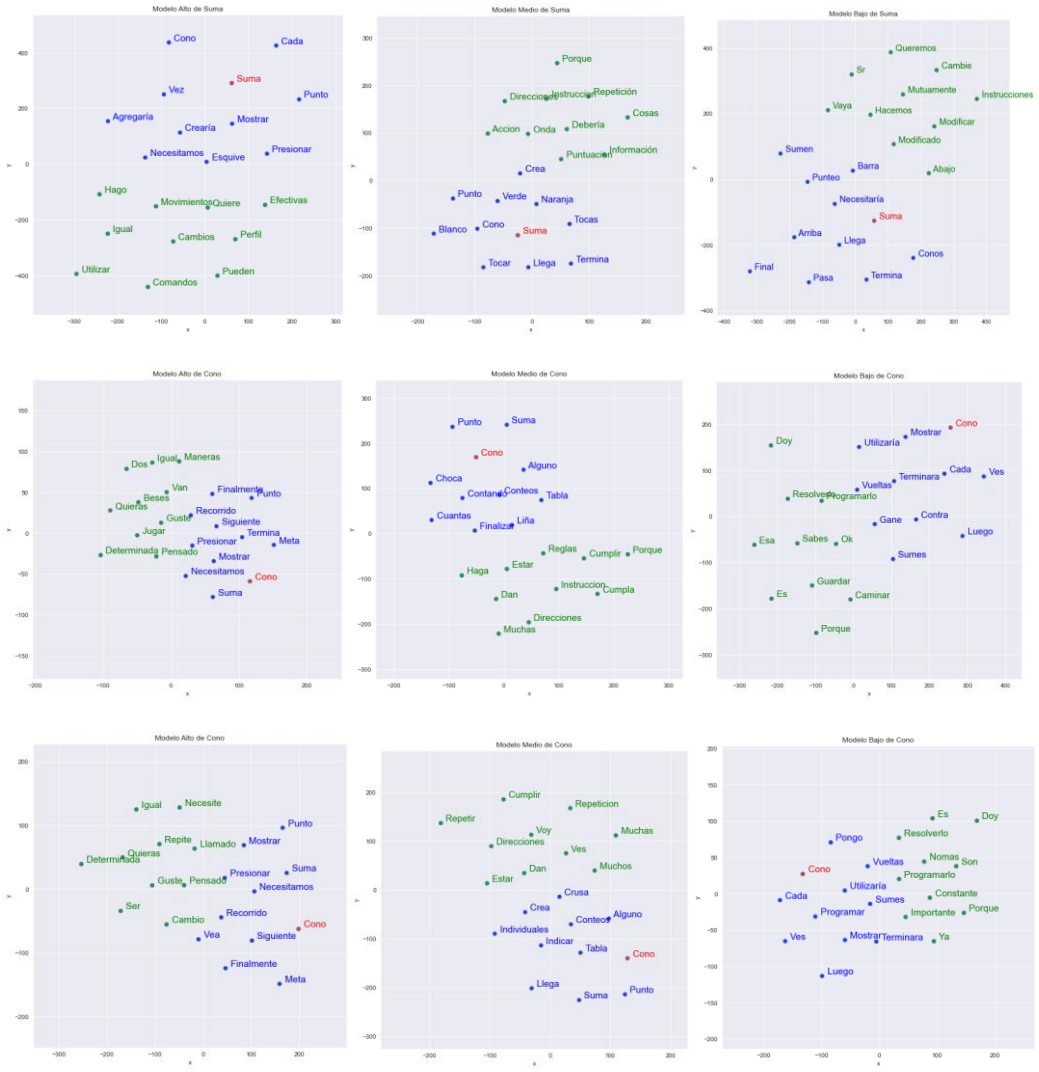
```
      numericInput(inputId = "n",
                   label= "Escriba el numero de cluster tomando en cuenta
el coeficiente silueta",
                   value = " ",
                   min = 2,
                   max = 6),
      tags$hr(),
      wellPanel(
      h5(helpText("Seleccione el tipo de parametros de su tabla")),
      radioButtons(inputId = 'sep', label = 'Separador', choices =
c(Coma=',',Punto_y_coma=';',Tab='\t', Espacio=''),
                   selected = ','),
      radioButtons(inputId = 'dec', label = 'Decimal', choices =
c(Coma=',',Punto='.'),
                   selected = ' ')
    ),

    br(),
    br()
    ),
    mainPanel(
      uiOutput("tb")

    )

  )
)
```

Otros resultados obtenidos con Word2Vec


Modelo Alto de Punto
Comandos
Es
Quieres
Utilizar
Movimientos
Perfil
Cambios
Debe
Cumpliría
Usa
Computadora
Crearía
Sumara
Esquive
Presionar
Auto
Necesitamos
Suma
Punto
Variable
Cono
Modelo Medio de Punto
Cosas
Direcciones
Programación
Marcas
Cumplir
Rdenes
Estar
Reglas
Mouse
Cierta
Crea
Alguno
Resolveria
Tabla
Jajaj
Cono
Toca
Toque
Punto
Suma
Blanco
Modelo Bajo de Punto
Puede
Después
Programarlos
Otra
No
Igual
Tenés
Dos
Marcador
Funciona
Situación
Encima
Gane
Empiece
Programado
Ultimo
Contra
Termine
Todo
Linea
Punto
Modelo Alto de Variable
Quieres
Maneras
Lo
Dos
Van
Comando
Efectivas
Quiere
Empezar
Hago
Deberíamos
Crearía
Ponerle
Sumara
Cuente
Variable
Llamada
Puntaje
Punto
Crear
Puntos
Modelo Medio de Variable
Afectar
Guardabas
Modificas
Cambias
Sume
Indica
Gun
Dirección
Resta
Estar
Leras
Cabezazo
Individuales
Contolaria
Posibilidades
Agregar
Victoria
Variable
Como
Podes
Puntos
Modelo Bajo de Variable
Lugar
Mismo
Contrario
Quedaría
Vuelve
Lo
Sucesivamente
Muevo
Ordenes
Fijo
Numeros
Ideas
Referirse
Usual
Utilizarla
Tipo
Almacenado
Creas
Cambiar
Variable
Valor
Modelo Alto de Variable
Punto
Variable
Crearía
Quiere
Ira
Crear
Cuente
Deberíamos
Puntos
Empezar
Diferente
Colocándola
Sumara
Quieres
Salto
Dicha
Llamada
Ponerle
Lo
Circulo
Ase
Modelo Medio de Variable
Puntos
Podes
Como
Agregar
Llame
Ayudar
Variable
Ver
Individuales
Utilizarla
Resta
Desbia
Facil
Estar
Lograr
Dirección
Mucho
Afectar
Programación
Modificas
Guardabas
Modelo Bajo de Variable
Cambiar
Variable
Crear
Ideas
Tipo
Vos
Numeros
Utilizarla
Marcadores
Creas
Empezar
Disfraces
Fijo
Puntos
Muevo
Ordenes
Vuelve
Lo
Haria
Mencionado
Hago
Modelo Alto de Color
Determinada
Pare
Situación
Múltiples
Naranjas
Tengo
Preciso
Poder
Acciones
Cantidad
Partida
Aparezca
Terminado
Terminaría
Finalmente
Desaparecer
Tocando
Pantalla
Color
Todo
Blanco
Modelo Medio de Color
Vez
Repetir
Aparece
Fútbol
Besos
Repeticion
Parecer
Puntuacion
Simple
Direcciones
Manejate
Sumas
Jajaj
Naranja
Blanco
Si
Rojo
Color
Verde
Suma
Tocas
Modelo Bajo de Color
Acción
Color
Bandera
Opciones
Basta
Actividades
Parar
Fijar
Usan
Estar
Conjunto
Mande
Objeto
Diferencia
Obtiene
Determinadas
Comportamientos
Va
Rayaría
Misma
Quieras


Modelo Alto de Color
Determinada
Cantidad
Guardar
Tengo
Ciertas
Vida
Reiniciaria
Color
Mouse
Necesita
Acciones
Repetición
Blanco
Todo
Finalmente
Aparezca
Poder
Fin
Tocando
Pantalla
Desaparecer
Modelo Medio de Color
Blanco
Tocas
Suma
Naranja
Color
Verde
Salga
As
Sumas
Manejate
Si
Vida
Beses
Jajaj
Rojo
Usarlo
Diferentes
Repetir
Direcciones
Aparecer
Vez
Modelo Bajo de Color
Color
Va
Objeto
Parar
Tocando
Rayaría
Estar
Obtiene
Bandera
Diferencia
Ir
Usan
Actividades
Quieras
Mande
Comportamientos
Haya
Simple
Movimientos
Modelo Alto de Auto
Pista
Destruye
Termina
Pasto
Auto
Conos
Tipo
Ese
Dicha
Distintas
Tocan
Teclas
Gato
Eje
Esquivar
Va
Veces
Los
Cuando
Ejemplo
Acción
Modelo Medio de Auto
Las
Flechas
Con
Osea
Conos
Tiempo
Destruye
Verde
Serian
Auto
Pongo
Me
Tocaba
Simpre
Cuenta
Cundo
Modelo Bajo de Auto
Veces
Repita
Repetir
Vas
Conos
Estructura
Llegando
Realiza
Los
Funcionar
Te
Decir
Terminara
Pondría
Game
Es
Over
Apareciera
Verde
Auto
Modelo Alto de Auto
Ese
Va
Porque
Cuando
Ninguna
Cumpliría
Veces
Beses
Tipo
Dicha
Esquivar
Contabilice
Mapa
Termina
Tocan
Destruye
Los
Auto
Conos
Pista
Punto
Modelo Medio de Auto
Me
Doy
Pongo
Tengo
Simpre
Diferentes
Cuenta
Tocaba
Destruye
Moveria
Hagas
Tiempo
Serian
Conos
Con
El
Auto
Osea
Verde
Las
Flechas
Modelo Bajo de Auto
Estructura
Das
Repetir
Decir
Vas
Algo
Muchas
Terminara
Misma
Los
Dibujaría
Repita
Es
Llegando
Game
Over
Pista
Auto
Conos
Modelo Alto de Tocar
Pasos
Algo
Pensar
Darse
Instrucción
Secuencia
Necesitábamos
Ejercicio
Cabezazo
Reiniciar
Moviera
Coordenadas
Final
Destruido
Salir
Cambiaría
Esconder
Llegar
Nuevamente
Verde
Tocar
Modelo Medio de Tocar
Debo
Suma
Liña
Funcione
Ciertos
Acaba
Simpre
Suceda
Puntuacion
Direcciones
Jajaj
Diferentes
Crea
Porque
Manejate
Instruccion
Accion
Naranja
Verde
Tocar
Termina
Modelo Bajo de Tocar
Haya
Perdiendo
Aparezca
Tocar
Ejercicio
Bloque
Problema
Bandera
Lenguaje
Opciones
Al
Sola
Nuevamente
Misma
Sintaxis
Ultimo
Empieza
Agrego
Lenguajes
Blanco
Veces


Modelo Alto de Tocar
Instrucción
Verde
Esconder
Trata
Tocar
Salir
Llegar
Nuevamente
Reiniciar
Algoritmo
Coordenadas
Detiene
Secuencia
Manteniendo
Darse
Especificas
Algo
Pensar
Sabes
Situación
Pasos
Modelo Medio de Tocar
Modelo Bajo de Tocar
Blanco
Aparezca
Ultimo
Tocar
Bloque
Opciones
Bandera
Perdiendo
Nuevamente
Parte
Agrego
Datos
Rápido
Creaste
Empieza
Lenguaje
Ejercicio
Sintaxis
Programa
Modelo Alto de Condición
Disfraces
Yo
Ganaste
Inicial
Entonces
Harían
Edificio
Ladrillos
Funcione
Cuenta
Comprobar
Condición
Falsa
Verdadera
Utilizaría
Hace
Foto
Das
Instrucción
Objetos
Cosa
Modelo Medio de Condición
Una
Misma
Esa
Restarlos
Voy
Ganaste
Instrucción
Llegas
Abrian
Aclarar
Pierda
Crusa
Condición
Empieza
Serian
Salirse
Acaba
Tener
Agaraste
Juego
Termina
Modelo Bajo de Condición
Tecla
Tal
Hasta
Usas
Pr
Introducción
Condición
Dar
Cumplir
Muere
Decide
Uno
Estar
Ranking
Así
Fondo
Poner
Objetos
Modelo Alto de Condición
Das
Instrucción
Cuenta
Tienes
Debes
Comprobar
Condición
Ganaste
Hace
Verdadera
Edificio
Más
Entonces
Funcione
Ladrillos
Reiniciaría
Yo
Foto
Disfraces
Reinician
Objetos
Modelo Medio de Condición
Tener
Esa
Salirse
Onda
Pierda
Usarlo
Niveles
Llegas
Abrian
Aclarar
Qures
Manejaría
Crusa
Ir
Voy
Serian
Sumas
Condición
Misma
Termina
Juego
Modelo Bajo de Condición
Objetos
Uno
Yo
Haría
Esteen
Explicaría
Así
Ranking
Nomas
Interrupción
Algunos
Muere
Pr
Dar
Casos
Introducción
Debemos
Hasta
Decide
Tecla
Condición
Modelo Alto de Suma
Cada
Dentro
Agregaría
Cambios
Utilizar
Movimientos
Mostrar
Suma
Presionar
Efectuada
Igual
Cono
Perfil
Esquive
Según
Comandos
Punto
Necesitamos
Guste
Vez
Más
Modelo Medio de Suma
Termina
Tocar
Suma
Porque
Verde
Sumas
Funciona
Naranja
Llegas
Debes
Repetición
Direcciones
Permite
Cumplir
Información
Tiene
Instruccion
Crea
Tocas
Blanco
Punto
Modelo Bajo de Suma
Sr
Cambie
Instrucciones
Modificarlo
Barra
Conos
Mutuamente
Queremos
Abajo
Llega
Final
Modificado
Vaya
Hacemos
Punteo
Necesitaría
Arriba
Suma
Do
Pasa
Termina

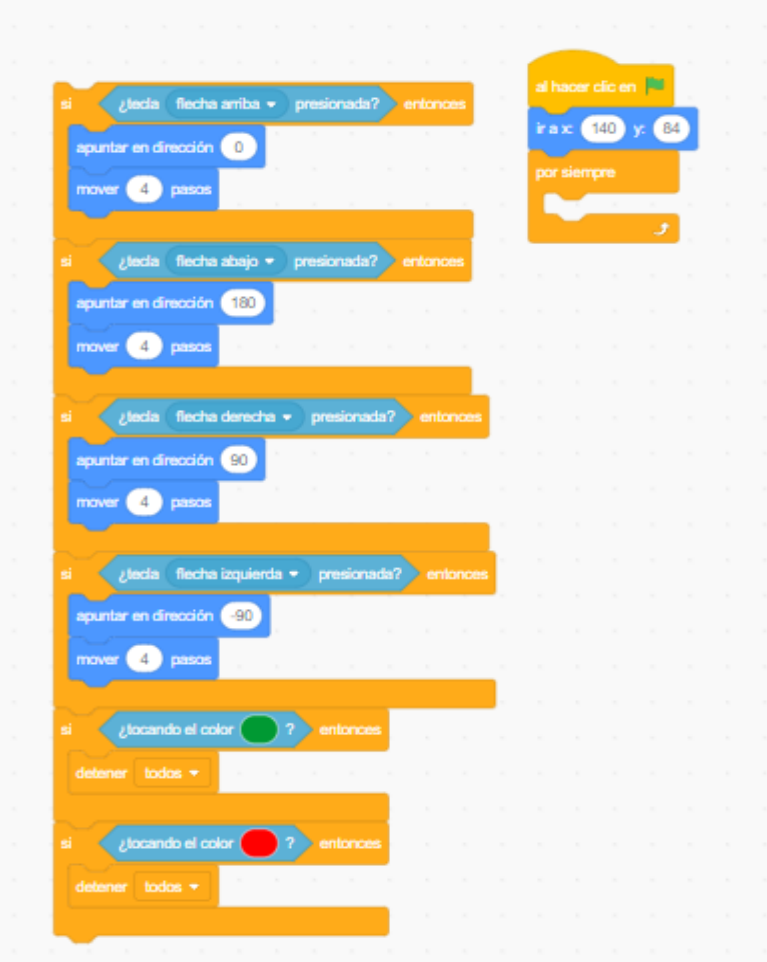


Ejemplos de programas y enunciados

Se presenta un código de ejemplo (en algunos de ellos no existe código para el cono), seguido de la descripción hecha por el estudiante sobre cómo explicaría la solución.

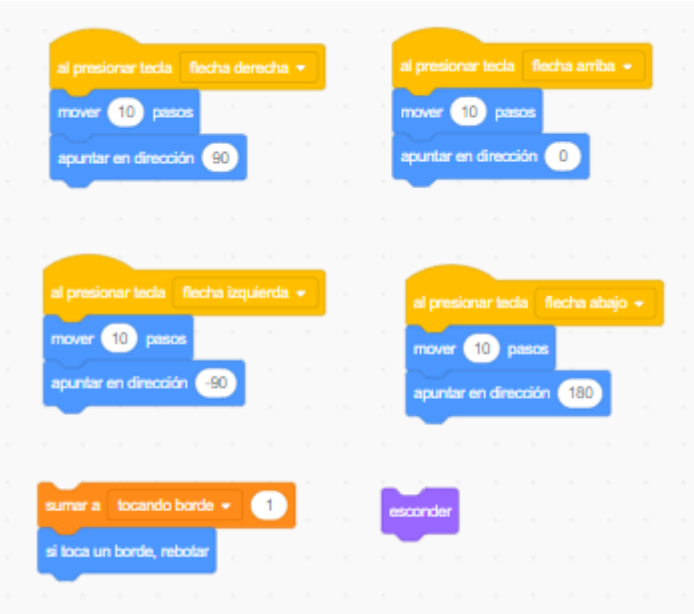
Calificación obtenida: es la calificación (redondeada en caso de ser necesario) por el programa construido.

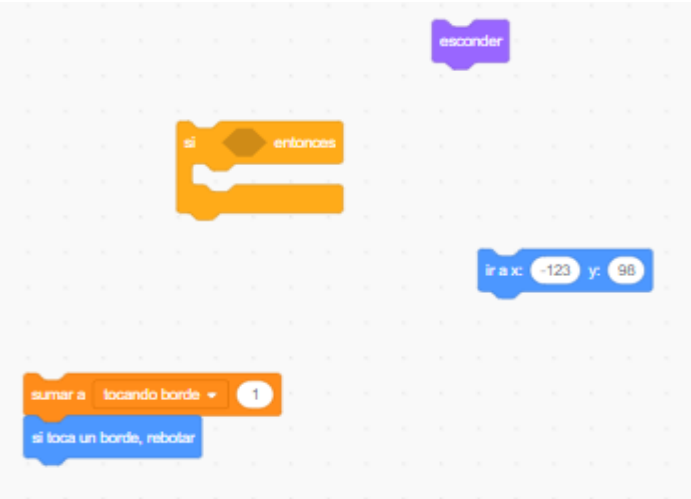
Puntaje obtenido: puntaje asignado para cada una de las cuatro categorías evaluadas (T_OBJETOS, T_MOVIMIENTOS, T_VARIABLES y T_CONDICIONES).



si el auto sale de la pista termina el juego,si tocas un cono te suma puntos y si llegas a la meta termina el juego osea ganaste

Calificación obtenida 8 - Puntaje obtenido 1 - 0 - 2 - 0

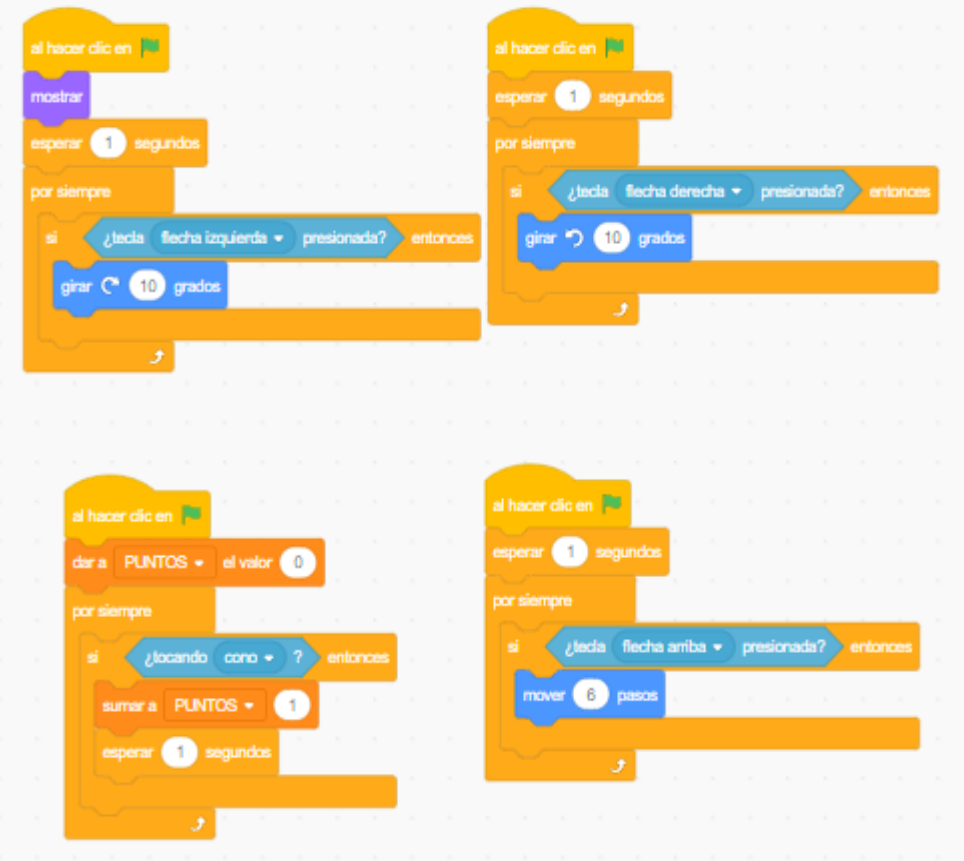


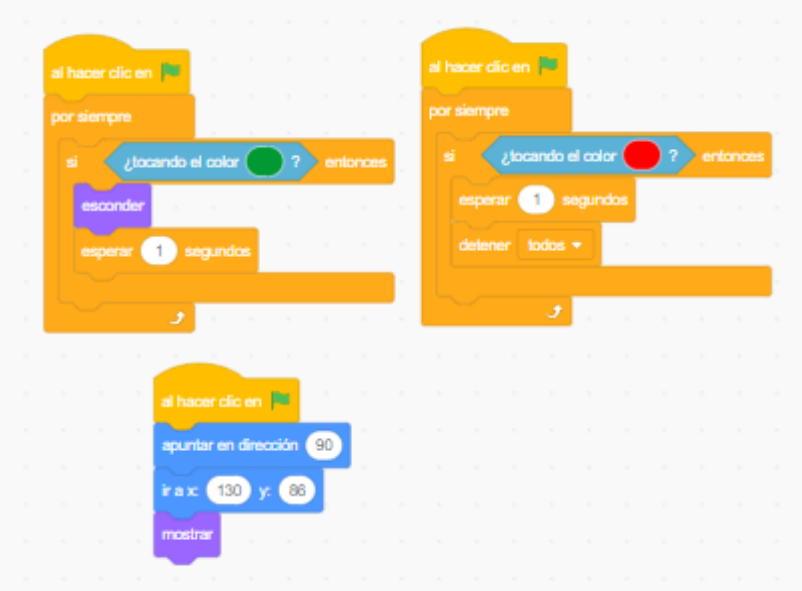


pondria el objeto auto en el escenario tambien los conos y la linea de meta luego dibujaria el escenario pintaria todo de verde y luego lo rayaria con el color blanco osuro luego pondria los objetos el cono\el auto\la linea de meta etc luego de ponerlos en su lugar empezaria a programar quiero que el auto valla rapido sin chocarse con ningun cono y llregando a la meta a los conos los progamaria que si el auto toca un cono apareciera 'game over' en la pantalla y terminara el juego y luego para la meta usaria una variable q llegabara el punteo de como voy

cuantos conos choque
cuantas veces pase la meta con mis punto
y cuantas veces voy perdiendo
y progamaria el fondo verde para que si el auto se sale de la pista apareciera 'game over' y acabaria el juego

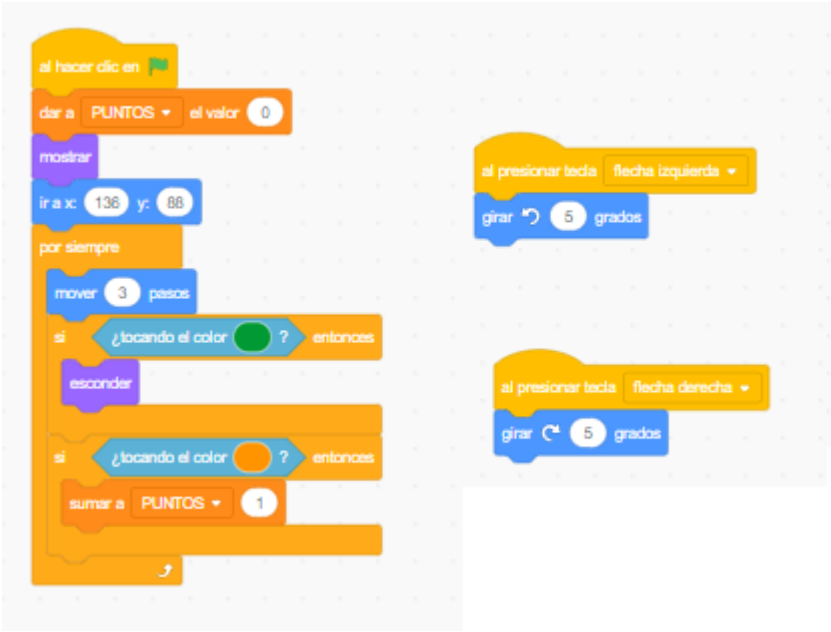
Calificación 3 - Puntaje obtenido 2 - 0 - 0 - 1





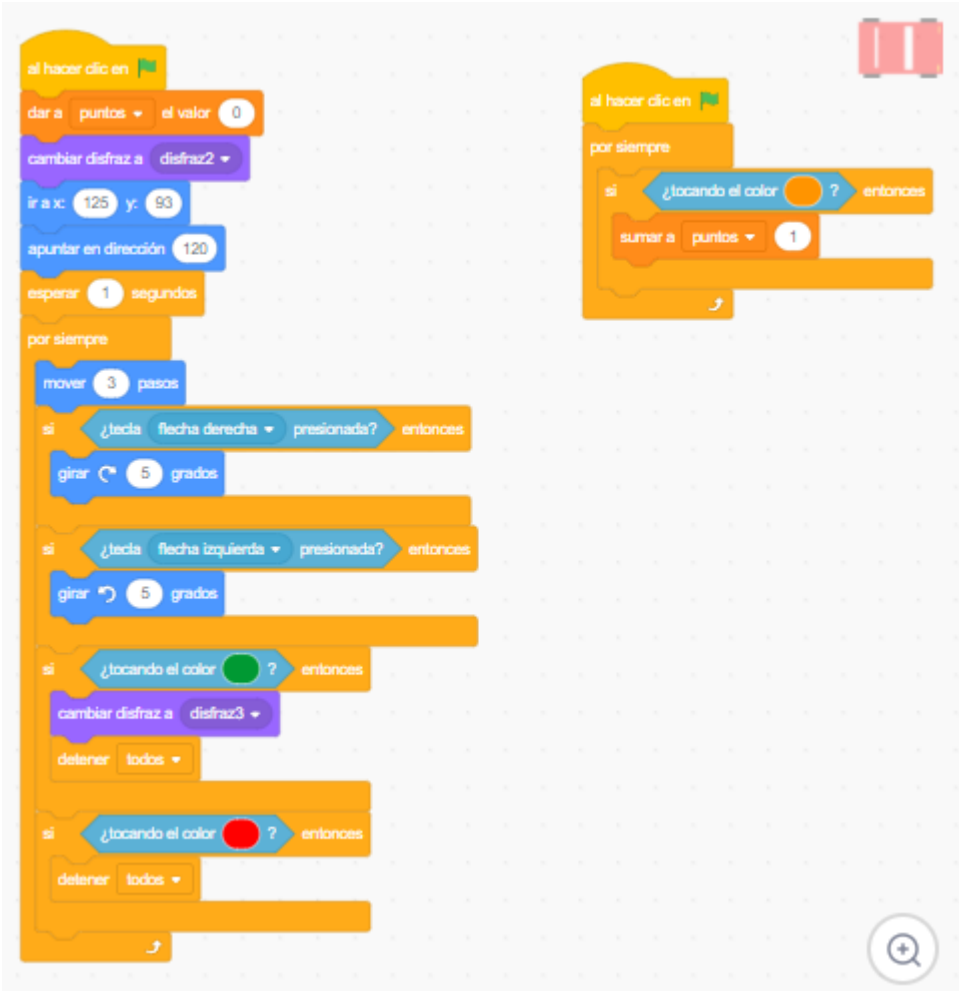
Yo el escenario lo hago igual que esta en la foto, utilizaría 6 objetos (5 conos y el auto) también para podes sumar los puntos creo una variable. Para hacer que el auto se destruya podría hacer que el auto desaparezca al salirse de la calle. Cuando se llega a la final podría poner un bloque que tenga "tocando tal color" cambie de escenario y diga fin del juego.

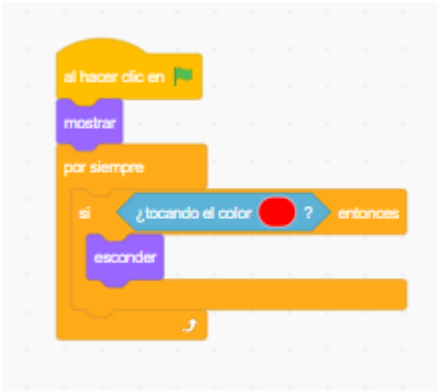
Calificación obtenida 10 - Puntaje obtenido 2 - 0 - 2 - 2



Abrian objetos que serian el auto los conos y el fondo .
El auto se moveria con las flechas cuando el auto toque el color verde se destruye.
Si el auto toca el color naranja suma un punto .
Al tocar el color blanco el juego termina .

Calificación obtenida 8 - Puntaje obtenido 1 - 2 - 1 - 2





tiene que tener un auto con 2 disfraces (objeto), una pista(fondo) y determinada cantidad de conos(objetos)
auto:programa

el auto se mueve siempre hacia delante, y se controla girando a la izquierda y derecha.
pero si toca el color verde cambia de disfraz.
también con una variable que tenga un valor inicial , y que se modifique cuando toque el objeto "cono".
finalmente al tocar color blanco "detener todo".

Calificación obtenida 12 - Puntaje obtenido 2 - 1 - 2 - 2